```
PPPPPPP     RRRRRRR    II    MM       MM     OOOOOO
PP     PP   RR     RR  II    MMMM   MMMM   OO      OO
PP     PP   RR     RR  II    MM MMM MM MM   OO      OO
PP     PP   RR     RR  II    MM  M  MM MM   OO      OO
PPPPPPP     RRRRRRR    II    MM     MM MM   OO      OO
PP          RR  RR     II    MM     MM MM   OO      OO
PP          RR   RR    II    MM     MM MM   OO      OO
PP          RR    RR   II    MM     MM MM    OOOOOO
```

BELSŐ LEIRÁS

Verzió: 84.1, Kiadva: 84.09.27

Összeállitotta: Détári György

Csak belső használatra !

TARTALOMJEGYZÉK

1. BEVEZETŐ

A PRIMO mikroszámitógép olcsóságánál fogva a házi számitógépek kategóriájába tartozik, de flexibilitása sok más célra is alkalmassá teszi. Ennek a leirásnak az a célja, hogy BASIC, Assembler vagy vegyes nyelvű programok irásához biztositsa a szükséges információkat. A könyv elsősorban kézikönyvként szolgál, ami azt jelenti, hogy egy-egy parancs, utasitás vagy terület leirásánál igyekszik teljes információt adni ahelyett, hogy más utasitásokra vagy fejezetekre utalna.

A könyvben sem az assembler, sem a BASIC nyelv definiciója nem szerepel. A BASIC nyelv leirásánál a "nyelvjárás" pontos megadása, nem pedig a BASIC programozás technikájának ismertetése volt a fő szempont.

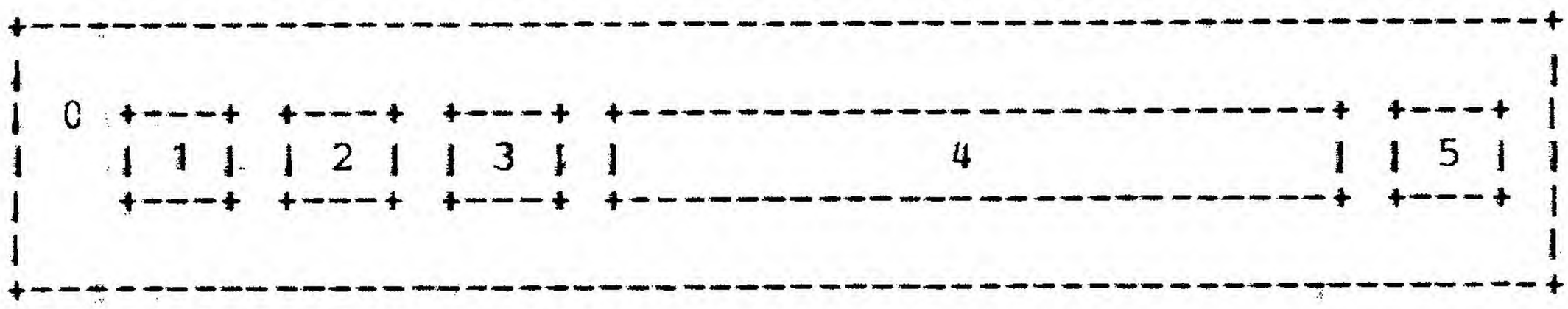
2. A PRIMO KONFIGURÁCIÓ ELEMEI

2.1 Alapgép

Az alapgép az NDK gyártmányú U880 processzorral működik, amely a ZILOG cég Z80 processzorával kompatibilis. A működtető órajel frekvenciája 2.5 MHz, de jumperrel 3.75 MHz-ra köthető. A tapasztalat szerint minden U880 processzor képes ezzel a magasabb sebességgel működni, de mivel a gyártó cég specifikációja 2.5, ezért a kikerülő gépek erre a frekvenciára vannak beállitva.

A beépitett memória 16 kbyte-os blokkokban bővithető. Az alapkiépités az A32, amelyben 16 kbyte ROM és 16 kbyte dinamikus RAM található. Az A48 és A64 tipusok az alapgéptől a RAM méretében térnek csak el. A ROM egy monitort (4 kbyte) és egy 12 kbyte-os BASIC interpretert tartalmaz. A hátul elhelyezett BUS csatlakozóra fel lehet dugni egy olyan dobozt (puttony), amely a ROM-ot lecseréli pl. egy Assembler programozási rendszerre.

Az alapgép hátoldalán találhatók a csatlakozók a kiegészitő berendezésekhez. A gép hátoldala (hátulról nézve):



Az ábra bal felső sarkában a RESET gomb látható. A csatlakozók tipusa és szerepe:

1 - kazettacsatlakozó tuchel (TAPE)

2 - COMMODORE BUS csatlakozó tuchel

3 - Joystick csatlakozó tuchel

4 - Z80 BUS kártyacsatlakozó

5 - Táp és TV csatlakozó speciális tuchel

2.2 Tápegység

Ez az egység a szükséges három tápfeszültségen kivül a TV csatlakozást is biztositja. A PS A-01 tipus TV modulátort tartalmaz az antennán át való csatlakozáshoz, mig a PS A-02 csak a videojel kicsatolását végzi. A tápcsatlakozó bekötése:

1 - Földelés

2 - Video kimenet (1 Volt PP, 75 Ohm)

3 - +12 Volt

4 - -5 Volt

5 - +5 Volt

2.3 Klaviatúra

A klaviatúra lényegében kapacitiv érzékelős, és 59 billentyűt tartalmaz. Ezeken szerepel a teljes magyar ABC kis és nagybetűkkel (néhány kivétellel), a számok, speciális jelek, és néhány vezérlőbillentyű (BRK, CLS, RETURN, SHIFT, UPPER, nyilak négy irányban, SPACE)

A klaviatúrán kivül még egy mikrokapcsolót is tartalmaz a

gép, ez a hátlapon elhelyezett RESET nyomógomb. A klaviatúra és a RESET kódolását software végzi. Az egyes gombok cimeit és kezelését lásd az Input/Output fejezetben, és az A. mellékletben.

2.4 Képernyő

A PRIMO-hoz bármilyen tipusú szines vagy fekete-fehér TV készülék csatlakoztatható, akár az antennacsatlakozón át (36-40 csatorna) a tápegységbe épitett modulátor segitségével, akár közvetlen videojel csatlakozással. A PRIMO a képernyőre kétféle fényerejű (sötét/világos) pontokat tud irni. A képernyő mérete egy jumpertől függően 256*256 vagy 192*256 lehet. A jelenlegi software csak az utóbbit támogatja. A képernyő frissitését autonóm hardware végzi a memória utolsó 6 illetve 8 kbyte-jából a jumpertől függően. A karaktereket a software bontja pontokra. A karakterkészlet az ASCII szabvány módositott (magyar betűket tartalmazó) változata. Bár a software karaktergenerálás lassúbb, mint a hardware, ugyanakkor sokkal rugalmasabb is, számos képernyőüzemmódot megenged. Az üzemmódváltásokat a kiirandó szövegben elhelyezett vezérlőkarakterek adják meg. A karakteres képernyőméret 16 sor * 42 pozició alapértelmezésben, mig ha átváltjuk az irás irányát függőlegesre, akkor 21 sor * 32 pozició lesz a méret (de a TV készüléket az oldalára kell fektetnünk). A képernyő alapszine lehet világos, ilyenkor a betűk sötéttel iródnak, de válthatjuk a karakterek alapszinét külön is. Mód van a szövegek aláhúzására, alsó és felső

index használatára is. A funkciók vegyesen is használhatóak. Lehetőség van kompozit karakterek létrehozására, mert alapértelmezésben a kiirás előtt a karakterpozició nem törlődik. A vezérlőkaraktereket a C. melléklet tartalmazza. A software lehetővé teszi a 128 feletti karakterkódok képeinek összeállitását, igy tetszőleges új karaktereket definiálhatunk. Az igy definiált karakterek nem feltétlenül egy sorosak, azaz pl. definiálhatunk 3-4 soros integráljelet is. A képernyő pontjai egyedileg gyújthatók, olthatók, igy egészen jó minőségű grafikus szolgáltatást kapunk. (ld. képernyőkezelés, grafikus függvények)

2.5 Hanggenerátor

A PRIMO-ba be van épitve egy kisteljesitményű hangszóró, amelynek kapcsaira programból lehet egyenfeszültséget kapcsolgatni. Igy egyszerű négyszöghullám generálására van módunk. A hanggenerátort a software a klaviatúra visszajelzésre és a kazettára irás indikálására használja. Programozását ld. az Input/Output fejezetben.

2.6 Kazettás magnó

A kazetta a PRIMO háttértára, ide lehet programokat, képernyőtartalmakat és adatokat menteni. Az információ a kazettára három egyenáramú szint programból történő váltogatásával irható fel (ld. Input/Output fejezet). A software a speciális PRIMO struktúráju file-ok irására és olvasására biztosit támcgatást (ld. kazettakezelés, kazettastruktúra), de az interface fizikai kezelésével bármilyen három szintet tartalmazó kazetta olvasható illetve irható. A kazettacsatlakozó DIN szabványos, és a készülék hátoldalán található. Bekötése:

1 - Kimenő jel (80 mV, 200 Ohm)

2 - Földelés

3 - Bejövő jel (1 mV, 4300 Ohm)

4 - Magnó vezérlés (+5 V)

5 - Magnó tápvezérlés (+5 V)

2.7 Nyomtató

A PRIMO-hoz lényegében bármilyen tipusú nyomtató kapcsolható, de a legalkalmasabbak azok, amelyek tartalmazzák a svéd karakterkészletet (Ilyen pl. a TERTA gyár COMPUTERTA tipusa). A csatlakoztatás a hátoldalon levő BUS csatlakozóra dugható illesztőkártyával történik.

2.8 Joystick

Ez az eszköz botkormányhoz hasonlit, amelyen egy külön gomb is található. Rendszerint játékprogramok alakzatait irányithatjuk segitségével, ilyenkor a külön gomb pl. a tüzelés. A joystick-be öt mikrokapcsoló van beépitve, egy a külön gombnak, négy pedig a különböző irányoknak. A botkormány elmozditása az iránynak megfelelően egy vagy két mikrokapcsolót zár, igy összesen nyolc irányt érzékelhetünk. A PRIMO -hoz két joystick kapcsolható. Az eszköz kezelését ld. az Input/Output fejezetben. A csatlakoztatás a hátlapon található DIN szabványos tuchel-en át történik. Bekötése:

1 - 1. joystick bejövő jel

2 - Földelés

3 - Joystick számláló léptető órajel

4 - 2. joystick bejövő jel

5 - Tápfeszültség (+5 V)

2.9 V24 interface

A magnetofon csatlakozó jumperrel átköthető, és ekkor a V24 szabványfelületnek megfelelő ± 5 Volt feszültséget adja ki, igy módunk van pl. távközlési vonal (MODEM) vezérlésére.

2.10 COMMODORE BUS

A hardware-ben minden elő van készitve COMMODORE perifériák, elsősorban floppydisk kezelésére. Az interface programozását ld. az Input/Output fejezetben. A szükséges jelek a hátlapon található DIN szabványos tuchel csatlakozón át adhatók ki. A csatolás Tri-state. A csatlakozó bekötése:

1 - Serv. Req

2 - Attn

3 - Serial data

4 - Földelés

5 - Serial Clock

3. BASIC NYELVLEIRÁS

3.1 Funkcionális billentyűk

CTR — Speciális kódok bevitelére (1-30) szolgál. Csak a kis/nagybetű billentyűkkel együtt nyomva használható. (Ld. B melléklet)

SHIFT — A kalviatúra felső állású jeleit aktiválja. Néhány egyéb karakterre is hat, ld. B melléklet.

UPPER — Megnyomása esetén a begépelt betűk nagybetűállásban kerülnek a gépbe. A funkciót a gomb újabb megnyomásával törölhetjük

RETURN — A sor végét jelzi bevitelnél (Ld. még EDIT parancs és programbevitel).

CLS — Ernyőtörlés

BRK — BREAK - A program, listázás ill. javitás befejezésére szolgál.

RESET — Feltétlen program ill. funkciómegállitásra szolgál. A gomb a PRIMO hátlapján található.

<-- — Visszalépés karaktertörléssel

SHIFT <-- — Sortörlés

: — BASIC utasitásszeparátor

. — A legutolsó aktiv (futás, listázás, javitás) sor sorszámát jelenti

? — A BASIC PRINT utasitás rövidítése

' — A BASIC REM utasitás rövidítése

3.2 BASIC változók

A BASIC változók neve az angol ABC betűivel kezdődő, és ugyanezen betűkkel vagy számokkal folytatódó karaktersorozat lehet. A nevekre fontos megkötés, hogy nem tartalmazhatnak BASIC alapszavakat, illetve nem egyezhetnek meg velük. Hosszú nevek használata nem célszerű egyrészt a nagyobb helyfoglalás miatt, másrészt mert a BASIC végrehajtás közben a változóneveket két karakterre csonkolja. Erre külön vigyázni kell, különben különböző neveken ugyanazt a változót használhatjuk hibajelzés nélkül. A változókat első előfordulásuk deklarálja. Kezdő értékük mindig nulla, illetve strinváltozókra üres string. A változók tipusa négy féle lehet. A tipust vagy postfix-el (a név után irt jelzőkarakterrel) vagy a kezdőbetű szerinti tipusdeklarációval (DEF... ld. később) adhatjuk meg. Tipusdeklaráció és postfix nékül a változó rövid lebegőpontos tipusú lesz. A különböző tipusú, de azonos nevű változókat a BASIC megkülönbözteti. Az egyes tipusok ábrázolása:

INTEGER Postfix jele a %, értéke -32768 -tól +32767 -ig terjedhet. Két byte-on, kettes komplemensben ábrázolódik: LSB, MSB, azaz elöl van az alacsony helyiérték. A logikai műveletek mindig integerek (egészek) között zajlanak, bitenkénti művelet formájában. A 0 érték a FALSE. A BASIC bármilyen nem nulla értéket TRUE-nak tekint. A reláció műveletek -1 -et (X'FFFF') használnak TRUE értéknek.

SINGLE Postfix jele a !, értéke a ±1.7E±38 tartományban

mozoghat, max. 7 decimális jegy pontossággal. Deklaráció és postfix nélkül ez a tipus az alapértelmezés. Négy byte-cn ábrázolódik: LSB, M, MSB, EXP azaz elöl van az alacsony helyiérték. Az M a közbülső byte-ot, az EXP a kitevőt jelenti. A szám alakja: mantissza * 2 ** kitevő. Az EXP = kitevő + 128. A mantissza abszolút értékben ábrázolódik, az MSB legelső bitje az előjel. A számok mindig normalizálva vannak, igy az első adatbit mindig 1, ezért ez nem is tárolódik. Az MSB első bitje az előjel után már a mantissza második bitje. A 0-t tiszta nulla jelzi.

DOUBLE Postfix jele a #, értéke a ±1.7E±38 tartományban mozoghat, max. 16 decimális jegy pontossággal. Nyolc byte-on ábrázolódik: LSB, M, M, M, M, M, MSB, EXP azaz elöl van az alacsony helyiérték. Az M a közbülső byte-ot, az EXP a kitevőt jelenti. A szám alakja: mantissza * 2 ** kitevő. Az EXP = kitevő + 128. A mantissza abszolút értékben ábrázolódik, az MSB legelső bitje az előjel. A számok mindig normalizálva vannak, igy az első adatbit mindig 1, ezért ez nem is tárolódik. Az MSB első bitje az előjel után már a mantissza második bitje. A 0-t tiszta nulla jelzi.

STRING Postfix jele a $, értéke 0-255 karakter lehet. Három byte-on ábrázolódik: HOSSZ, CIM/LSB, CIM/MSB. A tényleges string a tár tetején allokált külön területen van (ld. Memóriakiosztás és CLEAR parancs)

A változók leirása értékükkel együtt tárolódik a táblázatokban. A VARPTR speciális függvény (ld. ott) mindig a leirás után, az értékre mutat.

A változók más szempontból is csoportositathatók. Ábrázolásuk:

SKALÁR A leirás három byte-ból áll: TIPUS, NÉV/2., NÉV/1.
A tipus egyúttal az értékmező hossza is, azaz 2-INTEGER, 3-STRING, 4-SINGLE, 8-DOUBLE.

TÖMB Deklarációja kétféleképpen történhet; expliciten a DIM paranccsal, illetve impliciten, úgy hogy tömbként hivatkozunk rá. Az utóbbi esetben az egyes dimenziók a 0-10 tartományban deklarálódnak. A tömbméretet csak egyszer lehet deklarálni. A BASIC megkülönbözteti az azonos nevű tömböt és skalárt. A leirás változó hosszúságú, és a skalárisnál leirt három byte-tal kezdődik. Ezt követi két byte-on (LSB/MSB) a leiró folytatásának hossza. A következő byte a dimenziók száma, majd ezt a dimenziók leirása követi. Minden dimenzióleirás két byte (LSB/MSB), és a dimenzióhatár+1 -et tartalmazza. A leirás után jönnek az elemek értékei a skalárissal megegyező formában, sorfolytonosan (azaz a legutolsó index nő a leggyorsabban).

3.3 BASIC konstansok

±nnnn	Egész, valós, dupla konstans is megadható igy
±nn.nn	Valós és dupla kcnstans irható le igy
±nn.nnE±nn	Valós konstans megadása
±nn.nnD±nn	Dupla konstans megadása
"szöveg"	String konstans megadása. DATA utasitásnál, illetve bevitelnél a " elhagyható, ha a konstans elején nincs blank, vagy nem tartalmaz vesszőt. Üres string konstans is megengedett.

A konstansok tárolása a megadás formájától függ. A string-konstanst egyértelműen meghatározza az őt körülvevő kettősa-posztróf-pár. A numerikus konstansokra a BASIC a következő vizsgálatsort végzi el:

- A konstans duplapontos alakban tárolódik, ha
  - a szám nyolcnál több jegyű
  - normál alakos megadásnál az exponenst D betű jelzi benne
- A konstans rövid lebegőpontos alakban tárolódik, ha a szám nem duplapontos, és
  - tizedespontot tartalmaz
  - normál alakos megadásnál az exponenst E jelzi benne
  - értéke kivülesik a -32768,+32767 tartományon
- Ha a szám nem duplapontos, vagy rövid lebegőpontos, akkor egészként tárolódik.

3.4 Műveleti jelek, precedencia

+ Számokra összeadás, stringekre konkatenáció (összefűzés) művelet.

- Kivonás, vagy előjelváltás (unáris minusz)

***** Szorzás

/ Osztás

↑ Hatványozás (felfele nyil)

<,>,=,<=,>=,<>

Relációoperátorok. A relációoperátor kétváltozós függvénynek tekinthető, amely egész számot ad eredményül. A függvényérték -1, ha a reláció igaz, és 0, ha nem. A relációkat kifejezésekben és értékadó utasitásokban is használhatjuk, pl.

```
I% = (A <> B) + J%
```

A <> operátor >< formában is megadható. Megjegyezzük, hogy azonos stringek esetén a rövidebb a kisebb.

AND,OR,NOT Egész számok között végzi bitenként ÉS, VAGY illetve tagadás logikai műveleteket. Ha nem egész számok között áll, automatikus konverzió történik.

Két műveleti jel csak abban az esetben állhat egymás után, ha közülük a második az előjelváltást jelző minusz jel. Az egyes műveletek prioritása csökkenő sorrendben:

↑ Hatványozás (felfele nyil)

- Előjelváltás (unáris minusz)

***,/** Szorzás, osztás

+,-	összadás, konkatenálás, kivonás
<,>,=,<=,>=,<>	Relációk
NOT	Bitenkénti tagadás
AND	Logikai ÉS
OR	Logikai VAGY

Az azonos prioritású műveletek között a balról-jobbra szabály dönt. A prioritást tetszőleges mélységű zárójelezéssel változtathatjuk meg.

3.5 Automatikus tipuskonverziók

Tipuskonverzió csak a numerikus konstansok között történik. A konverzió a kiadott műveletektől függ:

+,-,* A műveletek eredményének tipusa megegyezik megegyezik a résztvevők közül legmagasabb pontosságú változó tipusával

/ Osztás esetén a művelet eredményének tipus megegyezik megegyezik a résztvevők közül legmagasabb pontosságú érték tipusával, azonban egész szám soha sem lehet. Egészek osztása esetén mindkét résztvevő rövid lebegőpontos számmá konvertálódik.

=,<,>,<=,>=,<> Relációmüveletek előtt a kisebb pontosságú résztvevő a nagyobb pontosságú résztvevőnek megfelelő tipusra konvertálódik.

AND,OR,NOT Logikai műveletek előtt a résztvevő(k) egész tipusúvá konvertálódnak. A konverzió miatt túlcsordulás lehetséges.

3.6 Programbevitel

Ha a gép alaphelyzetében egy sort úgy irunk be, hogy számmal kezdődik, akkor a sor programként tárolodik. A sorszám 0-65529 lehet. Ha a sor nem számmal kezdődik, akkor azt a BASIC azonnal végrehajtja. A leghosszabb program/adatsor legfeljebb 5 képernyősor (210 karakter) lehet. A BASIC a stringek belsejét kivéve a szóközöket szűri végrehajtáskor, igy, ha a program olvashatóságát feláldozva a sorokat felesleges szóközök nélkül gépeljük, jelentős helymegtakaritást érhetünk el.

A PRIMO BASIC bevitel közben nem elemzi szintaktikusan a szöveget, ezért egy program hibátlanságáról csak akkor lehetünk meggyőződve, ha annak minden utasitását legalább egyszer végrehajtottuk. A kulcsszavak és függvények neveit a BASIC viszont már beolvasás közben felismeri, és egyetlen byte-ként tárolja a helyfoglalás csökkentése céljából. A kódbyte-ok értéke nagyobb 128-nál. Mivel a ↓ (lefele nyil) bevitelkor speciális SHIFT gombként a kódértékeket X'40' -el növeli, igy az alapszavak egy karakterként is beirhatók.

```
+----------+
|          |
|   AUTO   |
|          |
+----------+
```

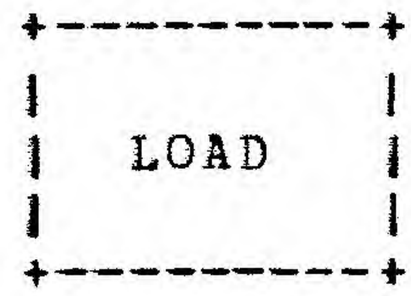
AUTO <nn<,nn>>

Automatikus programsorszámozást tesz lehetővé a beiráshoz. Az első szám a kezdősorszám, a második a lépésköz, mindkettő alapértelmezése 10. A funkcióból a BRK gombbal léphetünk ki. Ha már létező sorszámhoz érünk, a sorszám után * iródik (pl. 1010*). A RETURN gomb megnyomása a sort változatlanul hagyja. A RETURN a még nem létező sorokra kihagyott sorszámokat jelent. A CLS gomb törli az ernyőt, de a BASIC még mindig az előzőleg kiirt sor megadását várja.

```
+----------+
|          |
|  CLEAR   |
|          |
+----------+
```

CLEAR <nn>

A parancs hatására valamennyi változó elveszti korábbi értékét. Az nn paraméter a string terület (ld. Memóriakiosztás) méretét irja elő byte-ban. Ha nem adjuk meg, a méret változatlan marad. A BASIC indulásakor a string terület 50 byte hosszúra áll be. A parancs programból is hivható.



LOAD <"filenév<">>

Hatására a kazettáról az adott nevű illetve név hiányában a következő nem adat file betöltődik a memóriába. A keresés közben átlépett file-ok nevei kiiródnak a képernyőre SKIP: név formában. A file tartalmazhat BASIC vagy Assembler programot, illetve elmentett ernyőképet. Assembler program betöltése előtt a szükséges tárolóterületet a BASIC-től el kell venni (ld. Memóriakosztás - BASRAM leirás). A megfelelő BASRAM változót a parancsként kiadott POKE utasitással módosithatjuk. A névben szerepelhet speciális karakter is. A BASIC a kis és nagybetűket különbözőnek tekinti. A töltésről az ernyő utolsó sorában statisztika jelenik meg:

blckkszám hibaszám FOUND: név

A számok tizes rendszerben iródnak ki két jegyre. A funkció csak a RESET gombbal állitható meg. A parancs kiadható BASIC programból is. Ha assembler programot töltünk, és az nem indul automatikusan, a BASIC program visszakapja a vezérlést. Másik BASIC -et töltve a hivó program törlődik összes változójával együtt, és a betöltött program kap vezérlést. A két program fenntartott memóriaterületen át POKE és PEEK utasitásokkal adhat át adatokat egymásnak. A területet a BASIC-től el kell venni a megfelelő BASIC belső váltczó módositásával (ld. Memóriakiosztás - BASRAM leirás).

+----------+
| |
| TEST |
| |
+----------+

TEST <"név<">>

Hatására az adott nevű, illetve név hiányában a soron következő kazettafile olvashatóságát ellenőrzi. Az eredményről az ernyő utolsó sorában statisztika jelenik meg:

blokkszám hibaszám FOUND: név

A számok tizes rendszerben iródnak ki két jegyre. A funkció csak a RESET gombbal állitható meg.

+----------+
| |
| SAVE |
| |
+----------+

SAVE "filenév<">

Hatására a memóriában levő BASIC program az adott néven kazettára mentődik. A funkció csak a RESET gombbal állitható meg. A filenév speciális jeleket is tartalmazhat. A BASIC a kis és nagybetűket különbözőnek tekinti.

SAVE SCREEN "filenév<">

Hatására a képernyő pillanatnyi tartalma az adott néven kazettára mentődik. A funkció csak a RESET gombbal állitható meg. A filenév speciális jeleket is tartalmazhat. A BASIC a kis és nagybetűket különbözőnek tekinti.

CONT

Hatására a BRK gombbal, END vagy STOP utasitással leállitott BASIC program folytatja működését, ha ez lehetséges. A folytatást a program bármilyen módositása letiltja, azonban a programot listázni, a változókat kiiratni és/vagy módositani szabad. A megszakitott EDIT illetve LIST funkció nem folytatható.

DELETE sorszám

sorszám - sorszám

- sorszám

Hatására a kijelölt programsorok törlődnek. Az utolsó parancsforma a program elejétől töröl. A megadott sorszámoknak léteznie kell, különben a parancs nem hajtódik végre. Sorszám helyett adható a . (kurrens sor) szimbólum is.

EDIT sorszám

A megadott sorszámú BASIC programsort javithatjuk ezzel a funkcióval. Sorszám helyett adható a . (kurrens sor) szimbólum is. Hiba esetén, ha a BASIC a sort visszairja, azonnal

EDIT funkcióba kerülünk. Ha módositjuk a javitott sor sorszámát, az eredeti sor változatlan marad, és a javitott sor az új sorszámmal kerül a programba. Ha az új sorszám már létezik, felülirás történik. Az EDIT -et a következő gombokkal vezérelhetjük:

RETURN — Az EDIT végét jelzi. Ha az editálást RETURN -al kezdjük, a sor nem módosul. Megkezdett EDIT esetén a sor hátralevő része törlődik.

CLS — Ernyőtörlés, de a sor javitása folytatódik.

BRK — BREAK - Az EDIT funkció törlése, a sor nem módosul.

<-- — Visszalépés karaktertörléssel

--> — Egy karakter átmásolása az eredeti sorból

▼ — a lefele nyil gomb az EDIT újrakezdését jelenti az addigi javitások törlésével

SHIFT <-- — az EDIT újrakezdése, az addigi javitások maradnak

SHIFT --> — kilépés EDIT-ből a maradék sor átmásolásával

Egyéb — A javitott sor bővitése a beirt karakterekkel

+---------+
| |
| LIST |
| |
+---------+

LIST <sorszám>

<sorszám - sorszám>

< - sorszám>

<sorszám - >

Hatására a kijelölt programsorok kiiródnak az ernyőre. Sorszám nélkül megadva a funkció a teljes programot kiirja.

Sorszám helyett adható a . (kurrens sor) szimbólum is. A listázás csak addig folyik, mig valamilyen billentyűt nyomva tartunk. A BRK gomb a funkciót félbeszakitja. Ha egy sor listázásánál a megadott sorszám nem létezik, azonnal Ok üzenetet kapunk. Amennyiben határokat irunk elő, és a sorszámok nem léteznek, akkor az alsó határ helyett az azt követő első legnagyobb, felső határ helyett az azt megelőző legkisebb sorszámot veszi a parancs figyelembe. Üres tartomány listázására azonnali Ok a válasz.

LLIST <sorszám>

<sorszám - sorszám>

< - sorszám>

<sorszám - >

Hatására a kijelölt BASIC programsorok kiiródnak a nyomtatóra. Sorszám nélkül megadva a funkció a teljes programot kiirja. Sorszám helyett adható a . (kurrens sor) szimbólum is. Ha egy sor listázásánál a megadott sorszám nem létezik, azonnal Ok üzenetet kapunk. Amennyiben határokat irunk elő, és a sorszámok nem léteznek, akkor az alsó határ helyett az azt követő első legnagyobb, felső határ helyett az azt megelőző legkisebb sorszámot veszi a parancs figyelembe. Üres tartomány listázására azonnali Ok a válasz. A funkciót csak a RESET gombbal lehet leállitani.

```
+----------+
|          |
|   NEW    |
|          |
+----------+
```

NEW

Hatására a BASIC program teljes egészében törlődik.

```
+----------+
|          |
|   RUN    |
|          |
+----------+
```

RUN <nn>

Hatására a BASIC program elindul a megadott címkétől, illetve ennek hiányában a program legalacsonyabb sorszámú sorától. A RUN parancs törli valamennyi változó értékét, ezért újrainditás céljára a parancsként is kiadható GOTO nn utasitást használhatjuk. Ha a memórában nincs program, azonnali Ok választ kapunk.

```
+----------+
|          |
|   TRON   |
|          |
+----------+
```

TRON

Hatására a BASIC program futás közben érintett sorainak sorszáma kiiródik az ernyőre <cimke><cimke> ... formában. Programból is kiadható.

+---------+
| |
| TROFF |
| |
+---------+

TROFF

Felfüggeszti a TRON utasitás hatását. Programból is kiadható.

3.8 BASIC utasitások

3.8.1 Általános célú utasitások

+---------+
| |
| DEF... |
| |
+---------+

DEFINT kezdőbetű lista

DEFSNG kezdőbetű lista

DEFDBL kezdőbetű lista

DEFSTR kezdőbetű lista

Hatására azok a változók, amelyeknek kezdőbetűje a listán szerepel, postfix nélkül a megadott tipusúak lesznek (INT - egész, SNG - lebegőpontos, DBL - duplapontos, STR - string). A postfix megadása felülbirálja a DEF.. utasitás hatását. A lista állhat egyetlen elemből, vagy elemek vesszővel elválasztott felsorolásából. Egy elem vagy egy betű, vagy egy betűtartomány lehet. A tartományt az ABC rendben leirt betűk köze rakott elválasztójel jelzi, pl. DEFINT A,G-J,T Ha egy változó több DEF... listán szerepel, mindig az utolsó specifikáció lesz érvényes. A változótipusok többször is átdefi-

niálhatók, ilyenkor a korábbi tartalmaik a postfix megadásával érhetők el.

```
+----------+
|          |
|   DIM    |
|          |
+----------+
```

DIM név(n1<,n2<,n3 ...>>>)<,név(.....>

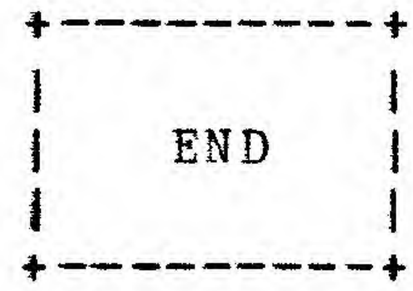
Az utasitás a tömbök dimenziószámának és dimenzióhatárainak leirására szolgál. A név a tömb neve, n1, n2 .. pedig az egyes dimenziók felső határa. Az alsó határ mindig nulla. A DIM utasitásban leirt tömbnév a DIM előtt nem szerepelhet a programban. A dimenziók helyén változó is állhat, ha a programfutás során a DIM utasitás előtt már értéket kapott.

```
+----------+
|          |
|   LET    |
|          |
+----------+
```

LET változó = kifejezés

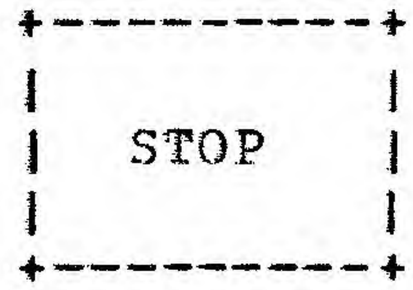
Az utasitás az aritmetikai értékadást jelöli. A LET alapszó elhagyható. A változó helyén állhat skaláris változó vagy tömbelem. A kifejezés változókból, tömbelemekből, konstansokból, függvényekből és a már felsorolt műveleti jelekből állhat. A BASIC a kiértékelés közben automatikus tipuskonverziót végez. A kifejezésben a string-ek mindig " jelek közé vannak fogva. A kifejezés a benne szereplő elemek tipusa szerint lehet aritmetikai, vagy stringkifejezés. A két kifejezéstipus nem vegyitető, azaz stringek és aritmetikai változók között nincs automatikus tipuskonverzió (természetesen aritmetikai kifejezés is tartalmazhat stringeket,

ha azck egy reláció két oldalán állnak).



END

Az utasitás a BASIC program futását leállitja. A program fizikai vége egyenértékű egy END utasitással.



STOP

Az utasitás a BASIC program futását leállitja, hatása azonos a BRK billentyűvel, azaz kiiródik a

Break in nn

üzenet. A program futása a CONT parancccsal folytatható. A STOP utasitást töréspontok elhelyezésére használhatjuk a programbelövés során. Megálláskor a változók tartalma kiiratható és/vagy megváltoztatható.

GOTO sorszám

Az utasitás feltétel nélküli ugrást okoz a megadott sorszámú programsorra. Utasitásként kiadva újraindithatjuk a programot anélkül, hogy a BASIC a változóértékeket törölné.

+----------+
| |
| GOSUB |
| |
+----------+

GOSUB sorszám

Az utasitás meghivja a megadott sorszámú programsorral kezdődő szubrutint (azaz feljegyzi a visszatérési cimet és ugrik a megfelelő programsorra). Utasitásként is kiadható, igy hibakeresésnél felhasználhatjuk a program után irt szubrutin hivására a töréspontokban. A szubrutin pl. kiirathatja a fontosabb változók értékét.

+----------+
| |
| RETURN |
| |
+----------+

RETURN

Az utasitás visszatér a GOSUB utasitással hivott szubrutinból. A programfutás a GOSUB-ot követő utasitással folytatódik.

+----------+
| |
| ON |
| |
+----------+

ON változó GOTO sorszám1<,sorszám2 ...>

Az utasitás hatására a BASIC a változó tartalma szerint választ a felsorolt sorszámlistából egy elemet. A változó értékét modulo 256 tekinti. Ha az értek 0, vagy nagyobb, mint a felsorolt sorszámok darabszáma, a következő utasitás hajtódik végre, egyébként feltétel nélküli ugrás történik a megadott sorszámú programsorra.

ON változó GOSUB sorszám1<,sorszám2 ...>

Az utasitás hatására a BASIC a változó tartalma szerint választ a felsorolt sorszámlistából egy elemet. A változó értékét modulo 256 tekinti. Ha az értek 0, vagy nagyobb, mint a felsorolt sorszámok darabszáma, a következő utasitás hajtódik végre, egyébként meghivja a megadott sorszámú programsorral kezdődő szubrutint (azaz feljegyzi a visszatérési címet és ugrik a megfelelő programsorra).

ON ERROR GOTO sorszám

Az utasitás hatására a BASIC feljegyzi a programsorszámot, és hiba esetén a vezérlést a megadott sorra adja. Ha sorszám helyett 0-t adunk, törli a korábban beállitott hibarutin címet.

+---------+
| |
| RESUME |
| |
+---------+

RESUME <sorszám>

<NEXT>

<0>

Csak akkor adható ki, ha a BASIC az ON ERROR utasitás hatására aktiválta a hibarutint. Az utasitás célja a hibarutinból való visszatérés. Ha a paraméter 0, vagy hiányzik, akkor a BASIC újrakezdi a hibázó utasitást. Ha a paraméter a NEXT szó, a vérehajtás a hibázó utasitást követő utasitáson folytatódik. Ha sorszámot adunk meg, a BASIC a kijelölt sorra ugrik. a RESUME utasitás törli a hibaállapotot.

ERROR

ERROR hibakód

Az utasitás az ON ERROR rutinok belövésére szolgál. A megadott kódú hibát szimulálja. A hibakódok a D. mellékletben találhatók. Ha érvénytelen hibakódot adunk meg, akkor a hiba a 20-as kódú UE (user error) lesz.

FOR

FOR változó = kifejezés1 TO kifejezés2

<STEP kifejezés3>

Cikluskezdet definiálása. A ciklust a NEXT utasitással kell zárni. A megadott ciklusváltozó kezdeti értéke az első kifejezés, végértéke a második kifejezés, mig a harmadik kifejezés a lépésközt adja meg. Ha a lépésközt nem adjuk meg, akkor a BASIC +1 -et használ. Ha a ciklusváltozó értékét a ciklusban csökkenteni akarjuk (azaz kifejezés1 > kifejezés2), akkor a negativ lépésközt (pl. STEP -1) meg kell adnunk. A ciklus egyszer mindenképpen lefut. Ha a lépésköz nulla, végtelen ciklus alakul ki. A BASIC a végértéket és a lépésközt a ciklus elején menti, igy ezek a ciklusmagban megváltoztathatók. Kerüljük azonban a ciklusváltozó átirását, mert ez kiszámithatatlan mellékhatásokkal járhat. Ciklusból csak akkor lépjünk ki, ha oda vissza is térünk.

+---------+
| |
| NEXT |
| |
+---------+

NEXT <változó1 <,változó2 ..>>

Ciklusvég definiálása. A ciklusváltozó(k) megadása nem kötelező, de ha megadtuk, egyeznie kell az éppen aktiv FOR utasitás változójával. Több ciklus közös végét adhatjuk meg a változók felsorolásával, ilyenkor az elsőként megadott változó a legbelső ciklushoz kell tartozzon. A programfutás során végrehajtott NEXT utasitás, függetlenül az elhelyezésétől, mindig a legutóbb megkezdett ciklushoz fog tartozni.

+---------+
| |
| IF |
| |
+---------+

IF egészkifejezés THEN utasitás1

<:ELSE utasitás2>

Feltételes szerkezet definiálása. Az egészkifejezés természetesen lehet reláció is. A relációk értéke 0 (FALSE), vagy -1 (TRUE) lehet. A BASIC minden 0-tól különböző értéket igaznak (TRUE) tekint. Ha a kifejezés igaz, végrehajtódik a THEN után irt utasitás1 rész, amely egy vagy több utasitás lehet. Az igaz ágat a sor vége, vagy az ELSE utasitás határolja be. Ha az utasitás egyetlen ugrás, akkor a THEN után elég a sorszámot megadni. Ebben az esetben az ELSE előtt felesleges a : megadása. Ha az ELSE ág szerepel, ennek hatóköre is a sor vége, és GOTO utasitás esetén itt is elegendő a sorszám megadása. A teljes IF-THEN-ELSE struktúra hossza legfeljebb egy programsor lehet. Egymásba skatulyázott fel-

tételes struktúrák esetén az ELSE mindig a legbelsó IF -hez tartozik. Ha ez nem felel meg, üres ELSE ágakat kell beiktatni.

```
+----------+
|          |
|   REM    |
|          |
+----------+
```

REM tetszőleges szöveg

Az utasitás kommentárok irására szolgál, és rövidithető aposztrófra. Mivel a REM utáni teljes sort a BASIC kommentárnak tekinti, a REM csak az utolsó utasitás lehet egy programsorban.

3.8.2 Adatátviteli utasitások

```
+----------+
|          |
|  PRINT   |
|          |
+----------+
```

PRINT lista<;TAB(n);lista ...>

Az utasitás a képernyőre való kiiratásra szolgál. A lista elemei lehetnek konstansok, változók, tömbelemek és kifejezések. Az elemeket egymástól vagy vesszővel, vagy pontosvesszővel választhatjuk el. A vessző hatására 16-os tabulácio történik, mig a pontosvessző elválasztás nélkül egymás mellé irja az elemeket (de a számok után mindig iródik egy szóköz). Az irás sorfolytonosan történik, azaz, ha a kurrens sor tele van, a következőben folytatódik. Ha az ernyő tele van, az irás mindig az utolsó sorba történik, mig az ernyőtartalom ugyanúgy csúszik felfele, mint az irógépben a

papir (SCROLL). A lista végén soremelés történik. Ha ezt el akarjuk kerülni, a lista végére pontosvesszőt kell irni. üres string kiirása esetén a kurrens pozició nem lép tovább. Speciális lehetőség a TAB(x) függvény, amely a következő elem kiiratási poziciójat vezérli. A TAB függvényekben megadott értékek 0-41 között lehetnek (a TAB csak az alsó hat bitet vizsgálja), és egy PRINT utasitáson belül monotonon növekedniük kell. A PRINT utasitásban ernyővezérlő karakterek is kiadhatók a listára irt CHR$ függvény segitségével. Az egész számok kiiratása a szokásos formában történik. A valós számok hét jegyig egész, efelett normál alakban iródnak ki. A duplapontos számok 16 jegyig iródnak ki egész formában, és efelett jelennek meg normál alakban. A pozitiv előjel helyett szóköz iródik ki.

PRINT# y,x,lista<;TAB(n);lista ..>

Az utasitas megegyezik a PRINT utasitással, azzal a különbséggel, hogy a kiirás az y,x pozición kezdődik. Az y 0-15, az x 0-41 lehet (illetve függőleges irás esetén 0-20,0-31, ld. a C. mellékletet).

PRINT USING "formátum";lista

A parancsnak ez a formája számok formátum szerinti kiiratását teszi lehetővé. A formátumot stringváltozóban is megadhatjuk. A formátum elemei:

#	A numerikus érték számjegyei. Közéjük keverve csak pont és vessző irható.
.	A tizedespont helyét jelöli ki.
,	Ha bárhol előfordul az érték számjegyei közt, a számjegyek hármas csoportokban, vesszővel elválasztva iródnak ki.

+	A mező előtt vagy után állhat. Kijelöli az előjel helyét, és megadja, hogy a pozitiv előjel + -ként iródjon ki.
-	A mező előtt vagy után állhat. Kijelöli az előjel helyét, és megadja, hogy a pozitiv előjel szóközként iródjon ki.
AAAA	A függőleges nyilak a lebegőpontos kitevő helyét adják meg.
**	A mező előtt állhat, és azt jelzi, hogy a vezető nullák helyett csillagok nyomtatódjanak.
₤₤	A mező előtt állhat, és azt jelzi, hogy a szám elé közvetlenül egy ₤ jel nyomtatódjék. A ₤ a szám és az előjele közé kerül.
**₤	A mező előtt állhat, és azt jelzi, hogy a vezető nullák helyett csillagok nyomtatódjanak, a szám elé közvetlenül pedig egy ₤ jel.
% %	Stringmezőt jelöl ki. A listán az ennek megfelelő pozición stringváltozónak vagy stringnek kell állnia.
!	A listán az ennek megfelelő pozición stringváltozónak vagy stringnek kell állnia. A ! hatására ennek első karaktere iródik ki.
Egyéb	Minden más beirt karakter szövegként kiiródik, és befejezi az egy számhoz tartozó mező leirását.

Ez az összetett utasitás jelentősen megkönnyiti a képernyőn a táblázatok összeállitását, mert igy szabályozhatjuk, hogy egy szám a nagyságától függetlenül mindig ugyanannyi poziciót foglaljon. Egy szám leirása legfeljebb 24 karakter lehet.

Ha a formátum túl hosszú, a maradék figyelmen kivül marad, míg ha lista hosszabb, mint a formátum, akkor a BASIC elölről kezdi a formátum értelmezését.

+---------+
| |
| LPRINT |
| |
+---------+

LPRINT lista<;TAB(n);lista ...>

Az utasitás megegyezik a PRINT utasitással, azzal a különbséggel, hogy a kiirás a nyomtatóra történik, és csak az X'0D', X'0A' - soremelés, X'0C' - lapdobás vezérlőkódok hajtódnak végre.

LPRINT USING "formátum";lista

Az utasitás megegyezik a PRINT USING utasitással, azzal a különbséggel, hogy a kiirás a nyomtatóra történik.

+---------+
| |
| INPUT |
| |
+---------+

INPUT <"szöveg";>lista

Az utasitás a klaviatúráról olvas konverzióval. Kiadásakor kiiródik a szöveg, majd egy kérdőjel, és megjelenik a cursor. A lista lehet egy vagy többelemű, string és aritmetikai változók egyaránt szerepelhetnek rajta. A válaszadást többféleképpen végezhetjük, vagy begépeljük az összes értéket vesszővel elválasztva, vagy egyesével adjuk meg őket. A bevitelt mindig RETURN-al jelezzük. Ha egy stringváltozóba blank értéket is be akarunk irni, akkor kettősaposztrófok közé zárva kell megadnunk. Ha elemenkénti beadás van, az

utasitás az újabb elemeket két kérdőjellel kéri. Ha hibás adat került be, kiiródik a ?REDO szöveg, és ilyenkor az egész listát elölről be kell irni. Ha egyszerre irtuk be a listát, és túl sok elemet adtunk meg, akkor az ?EXTRA IGNORED üzenet jelenik meg, és a fölösleges elemeket az utasitás figyelmen kivül hagyja.

```
+---------+
|         |
| INKEY$  |
|         |
+---------+
```

stringváltozó=INKEY$

Ez a függvény egy egykarakteres stringet ad eredményül. Ha a klaviatúrán le van nyomva egy gomb, akkor ennek értékét kapjuk meg, mig ha nincs, akkor az eredmény az üres string. Az utasitás hatására a cursor nem jelenik meg, a beirt karakter pedig nem iródik ki az ernyőre.

```
+---------+
|         |
|  DATA   |
|         |
+---------+
```

DATA lista

Az utasitás adatmező leirásra szolgál, amelyet aztán a READ utasitással helyezhetünk el tömbökbe és változókba. A lista elemei lehetnek fix és lebegőpontos számok, valamint stringek is. A stringeket nem kell aposztrófok közé zárni, csak akkor, ha vezető szóközt vagy vesszőt tartalmaznak. A programba bárhol beirt DATA utasitásokat a BASIC egyetlen nagy tömbként kezeli.

```
+---------+
|         |
|  READ   |
|         |
+---------+
```

READ lista

A DATA utasitás(ok)kal leirt adatmező tömbökbe/változókba való elhelyezésére szolgál. Elvégzi a tipusegyeztetést, és a szükséges konverziót. Az olvasást a legelső DATA utasitásnál kezdi, majd minden READ az előző READ poziciójától folytatja. Az olvasás helyét befolyásolhatjuk a RESTORE utasitással. Ha több adatot akarunk olvasni, mint amennyi a DATA mezőben szerepel, OD (Out of Data) hibát kapunk.

```
+---------+
|         |
| RESTORE |
|         |
+---------+
```

RESTORE <sorszám>

Hatására a következő READ utasitás a feldolgozást a megadott sorszámú DATA utasitástól kezdi. Ha a sorszámot nem adtuk meg, a feldolgozás a DATA mező elejétől kezdődik.

3.8.3 Kazettakezelő utasitások

```
+---------+
|         |
|  OPEN   |
|         |
+---------+
```

OPEN string

Az utasitás a megadott string vagy stringváltozónak megfelelő adatfile cimkét kikeresi a kazettán. Ha a stringet nem

adjuk meg, a soron következő adatfile-t nyitja meg.

+----------+
| |
| CREATE |
| |
+----------+

CREATE string

Az utasitás a megadott string vagy stringváltozónak megfelelő file cimkét felirja a kazettára.

+----------+
| |
| PRINT# |
| |
+----------+

PRINT# lista

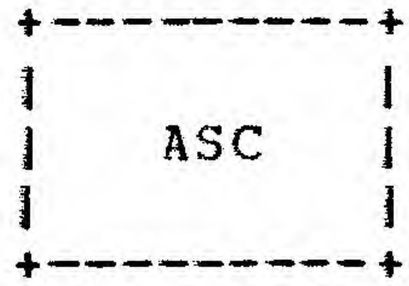
Az utasitás megegyezik a PRINT utasitással, azzal a különbséggel, hogy a kiirás a kazettára történik.

+----------+
| |
| INPUT# |
| |
+----------+

INPUT# lista

Az utasitás a kazettáról olvas konverzióval. A lista lehet egy vagy többelemű, string és aritmetikai változók egyaránt szerepelhetnek rajta. A lista végén a BASIC új sor jelet vár, tehát a lista formátumának szigorúan egyeznie kell a kazettaformátummal. Adathiba, vagy túlolvasás esetén a program hibajelzéssel leáll.

3.8.4 String függvények

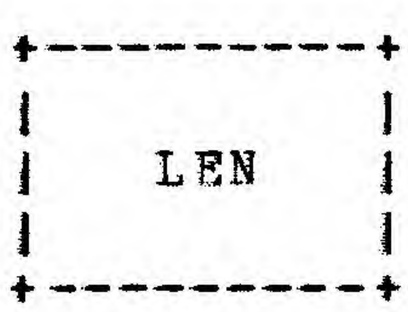


ASC (string)

A függvény a megadott string vagy stringváltozó első karakterének kódértékét adja.

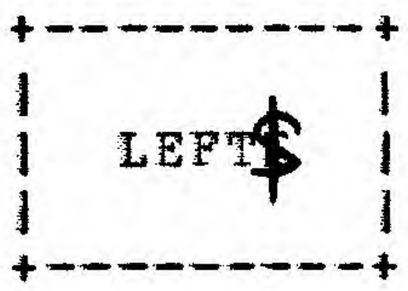
CHR$ (kifejezés)

A függvény a megadott kifejezés értékének megfelelő kódértékű karaktert adja.



LEN (string)

A függvény a megadott string vagy stringváltozó kurrens hosszát adja.



LEFT$ (string,n)

A függvény a megadott string vagy stringváltozó bal szélső n karakterét adja. Az n helyén kifejezés is állhat.

RIGHT$

RIGHT$ (string,n)

A függvény a megadott string vagy stringváltozó jobb szélső n karakterét adja. Az n helyén kifejezés is állhat.

MID$

MID$ (string,n,m)

A függvény a megadott string vagy stringváltozó n. -dik poziciójától m karaktert ad. Az n helyén kifejezés is állhat. A poziciók számozása 1 -től kezdődik.

STRING$

STRING$ (n,m)

Az m helyén kettősaposztrófok között állhat egy karakter, vagy megadható a karakter kódértéke is kifejezés formájában. A függvény az m -el megadott karaktert n -szer ismételve állit elő új stringet.

```
+---------+
|         |
|  STRE   |
|         |
+---------+
```

STRE (kifejezés)

A függvény a megadott kifejezést a PRINT utasitásnak megfelelő formában karakteresre konvertálja. A kapott string a függvény értéke, amely abban különbözik csak a PRINT által nyomtatott képtől, hogy a végén nincs elválasztó blank.

```
+---------+
|         |
|   VAL   |
|         |
+---------+
```

VAL (string)

A függvény a megadott string első vagy egyetlen numerikus mezejét számmá konvertálja. Ha a stringnek nincs numerikus része, az eredmény nulla lesz.

3.8.5 Matematikai függvények

```
+---------+
|         |
|   ABS   |
|         |
+---------+
```

ABS (x)

A függvény a megadott x kifejezés abszolút értékét adja.

```
+---------+
|         |
|   ATN   |
|         |
+---------+
```

ATN (x)

A függvény a megadott x kifejezés Arcus Tangens -ét adja radiánban.

```
+---------+
|         |
|  CDBL   |
|         |
+---------+
```

CDBL (x)

A függvény a megadott x kifejezést dupla pontossággal számolja ki.

```
+---------+
|         |
|  CINT   |
|         |
+---------+
```

CINT (x)

A függvény azt a legnagyobb egésszámot adja, amely a megadott x kifejezésnél kisebb (Azaz CINT(-0.5)=-1, CINT(0.5)=0). Az eredmény egész.

```
+---------+
|         |
|   COS   |
|         |
+---------+
```

COS (x)

A függvény a megadott x kifejezést radiánnak tekintve a kifejezés Cosinus -át adja.

```
+---------+
|         |
|  CSNG   |
|         |
+---------+
```

CSNG (x)

A függvény a megadott x kifejezés értékét rövid lebegőpontos számmá konvertálja.

```
+---------+
|         |
|   EXP   |
|         |
+---------+
```

EXP (x)

A függvény a megadott x kifejezés exponenciális (e**x) függvénye.

```
+---------+
|         |
|   FIX   |
|         |
+---------+
```

FIX (x)

A függvény a megadott x kifejezésből törli a tizedespont utáni számjegyeket, és az eredmény az igy kapott egésszám. (Azaz FIX(-0.5)=0, FIX(0.5)=0)

```
+---------+
|         |
|   INT   |
|         |
+---------+
```

INT (x)

A függvény azt a legnagyobb egésszámot adja, amely a megadott x kifejezésnél kisebb. Az eredmény lebegőpontos. (Azaz INT(-0.5)=-1, INT(0.5)=0)

```
+---------+
|         |
|   LCG   |
|         |
+---------+
```

LOG (x)

A függvény a megadott x kifejezés tizesalapú logaritmusát adja.

```
+---------+
|         |
|   LN    |
|         |
+---------+
```

LN (x)

A függvény a megadott x kifejezés természetes (e) alapú logaritmusát adja.

```
+---------+
|         |
|   RND   |
|         |
+---------+
```

RND (x)

A függvény véletlen számokat generál. Ha x helyén 0 szerepel, akkor 0 és 1 közé eső valós számokat kapunk, amelyek közt a két határ nem fordul elő. Ha x helyén pozitiv szám áll, akkor a generátor 1 és x közé eső egész számokat generál, amelyek között a két határ előfordulhat.

```
+---------+
|         |
| RANDOM  |
|         |
+---------+
```

RANDOM

Ha az RND(X) függvény használatakor a program két különböző futásához különböző értékek szükségesek, a RANDOM függvény segitségével az RND(X) kezdőértéke véletlenszerűen megváltoztatható.

```
+---------+
|         |
|   SGN   |
|         |
+---------+
```

SGN (x)

Az x kifejezés előjel függvénye, azaz pozitiv x-re 1, negativra -1, nullára 0 egésszámot ad eredményül.

```
+---------+
|         |
|   SIN   |
|         |
+---------+
```

SIN (x)

A függvény a megadott x kifejezést radiánnak tekintve a kifejezés Sinus -át adja.

```
+---------+
|         |
|   SQR   |
|         |
+---------+
```

SQR (x)

Az x kifejezés négyzetgyökét adja meg.

```
+---------+
|         |
|   TAN   |
|         |
+---------+
```

TAN (x)

A függvény a megadott x kifejezést radiánnak tekintve a kifejezés Tangens ét adja.

3.8.6 Speciális függvények

```
+---------+
|         |
|   ERL   |
|         |
+---------+
```

ERL

A függvény a gép bekapcsolása óta utoljára hibát okozó sor sorszámát adja meg. Ha hiba még nem volt, 0-t ad vissza, illetve assembler rutinból hivva -32767 -et.

```
+---------+
|         |
|   ERR   |
|         |
+---------+
```

ERR

A függvény a legutoljára elkövetett hiba kódját adja meg.

A D. mellékletben felsorolt hibakódokat az

ERR/2+1

kifejezésből kaphatjuk meg.

```
+---------+
|         |
|   POS   |
|         |
+---------+
```

POS(0)

A függvény a cursor kurrens pozicióját (0 - 41) adja meg a display kurrens során belül.

```
+---------+
|         |
|   FRE   |
|         |
+---------+
```

FRE (skaláris változó)

A függvény a megadott változó tipusától függően a szabad memória, illetve szabad stringmemória méretét adja byte-okban.

```
+---------+
|         |
|   INP   |
|         |
+---------+
```

INP (x)

A függvény az x (0-255) című port-ról beolvas egy byte-ot (ld. Input/Output fejezet).

```
+---------+
|         |
|   OUT   |
|         |
+---------+
```

OUT x,b1<,b2...>

A függvény az x (0-255) című port-ra kiküldi a megadott adatscrt (ld. Input/Output fejezet).

```
+----------+
|          |
|   PEEK   |
|          |
+----------+
```

PEEK (x)

A függvény az x című memóriabyte-ot adja eredményül. Ha a cim nagyobb, mint 32767, akkor helyette a 65536-x értéket kell megadni.

```
+----------+
|          |
|   POKE   |
|          |
+----------+
```

POKE x,b1<,b2...>

A függvény az x című memóriabytetól betölti a megadott adatsort. Ha a cim nagyobb, mint 32767, akkor helyette a 65536-x értéket kell megadni.

```
+----------+
|          |
|  VARPTR  |
|          |
+----------+
```

VARPTR (x)

A függvény az x nevű változó vagy tömbelem cimét adja meg.

3.8.7 Hanggenerátor kezelés

BEEP x,n<;x,n ....>

Ez az utasitás a hanggenerálást segiti. Ellentétben pl. a klaviatúra leütéskor keletkező hangokkal, ez kimegy a magnócsatlakozóra, igy felvehető, illetve erősitőre vezethető.

Az x érték a fél rezgésidőt adja meg a következő módon:

T/2=8.26*x+36.44 usec

A generált hangok tartománya 400Hz-18kHz lehet. Az n értéke a félhullámok számát, azaz a hang időtartamát adja meg.

3.8.8 Grafikus függvények

Megjegyezzük, hogy a képernyő az itt felsorolt parancsok és függvények használata nélkül is kezelhető a PEEK/POKE függvényekkel. A képernyőpuffer a memória utolsó 6 kbyte részét foglalja el, igy cime a PRIMO memóriaméretétől függ:

PRIMC A32 26624

PRIMC A48 43008 => -22528

PRIMC A64 59392 => -6144

A képernyő nyolc vizszintesen egymás mellett fekvő pontja felel meg egy byte-nak. A puffer kezdőcime a képernyő bal felső sarkának felel meg. A puffer byte-jait a PEEK/POKE függvényekkel manipulálva tetszőleges ábra létrehozható. Az 1-es értékű bitek mindig világosak lesznek. Mivel vizszintes

irányban 256 pont van, egy pontsornak 32 byte felel meg. Az ernyő 192 pontsort tartalmaz.

```
+----------+
|          |
|   SET    |
|          |
+----------+
```

SET (x,y)

A parancs a képernyő x,y koordinátájú pontját a képernyő alapszinével (sötét/világos) ellentétesre váltja. A (0,0) pont a képernyő bal alsó sarka, a címzés nullától indul. Az x= 0-255, az y= 0-191 tartományban adható meg.

```
+----------+
|          |
|  RESET   |
|          |
+----------+
```

RESET (x,y)

A parancs a képernyő x,y koordinátájú pontját a képernyő alapszinével (sötét/világos) egyezőre váltja. A (0,0) pont a képernyő bal alsó sarka, a címzés nullától indul. Az x= 0-255, az y= 0-191 tartományban adható meg.

```
+----------+
|          |
|  POINT   |
|          |
+----------+
```

POINT (x,y)

A függvény a képernyő x,y koordinátájú pontját teszteli. Ha a pont értéke a képernyő alapszinével (sötét/világos) egyezik, a függvényérték 0, ellenkező esetben -1 lesz. A (0,0) pont a képernyő bal alsó sarka, a címzés nullától

indul. Az x= 0-255, az y= 0-191 tartományban adható meg.

```
+---------+
|         |
|   CLS   |
|         |
+---------+
```

CLS

A parancs a képernyőt törli, és a kurrens poziciót a bal felső sarokba helyezi.

3.8.9 Assembler rutinok hivása

```
+---------+
|         |
|  CALL   |
|         |
+---------+
```

CALL (cim <,egész paraméter>)

Ez a funkció a megadott cimű assembler programot inditja. Ha a cim nagyobb, mint 32767, akkor helyette a 65536-cim értéket kell megadni. Ha az opcionális paramétert megadjuk, akkor az assembler program ezt a DE regiszterpárban kapja meg.* Assembler szubrutin elhelyezésére többféle lehetöség is adódik.

- A BASIC program tartalmazza az assembler rutint számok formájában. Ebben az esetben irhatjuk a rutint DATA utasitásba, és innen tömbbe olvasva, vagy a memória szabad területeire a POKE függvénnyel betöltve használhatjuk. A POKE VARPTR(tömb(0)),x,x,x.. sorozattal közvetlenül is tömbbe helyezhetjük a programot. A tömbben futó programokat önáthelyezésre kell irni, mert a BASIC a tömb kezdőcimét minden új skaláris változó felvételekor módositja. A legegyszerűbb

* A visszaadandó értéket a HL-be kell vinni. Belépéskor BC = return address (stack-en), DE = param, HL = belépési pont

módszer az, ha a programban csak relativ ugró utasitások vannak, és az adatterületet a BASIC tartalmazza. A program hivása a CALL (VARPTR(tömb1(0)),VARPTR(tömb2(0))) utasitással történhet, ahol a tömb1 a programot, a tömb2 az átadott paramétereket és munkaterületet tartalmaz. A főprogram címét HL-ben kell átadnia az ASM programnak

- Az assembler programot vagy kézzel, vagy BASIC-ből kazettáról töltjük.

Az assembler rutinokra vonatkozólag ld. a BASRAM leirást, és a szalagformátum leirást. Megjegyezzük, hogy az assembler rutinok hiváskor minimálisan 58 byte hosszú STACK-et kapnak.

Ha a BASIC-től területet veszünk el a programnak, akkor az a LOAD funkcióval betölthető mind a BASIC-ből, mind parancsként kiadva. A nem automatikus inditású program ezután CALL utasitással hivható (parancsként pl. PRINT CALL(cim), vagy I%=CALL(cim)). A RUN parancs mindig a BASIC programot inditja.

Fontos ! A rendszer az IX regisztert beállitja a DCB címére, ezt nem szabad rontani, ha a rendszerrutinokat használni akarjuk ! Az IY regiszter, és a második regiszterkészlet szabadon használható.

4. MEMÓRIA KIOSZTAS

```
X'0'      +-----------+
          |  MONITOR  |   4 kbyte MONITOR (SYSTEM) ROM
X'1000'   +-----------+
          |  BASIC    |   12 kbyte BASIC ROM
X'4000'   +-----------+
          |  SYSRAM   |   SYSTEM RAM terület
X'407F'   +-----------+
          |  BASRAM   |   BASIC RAM terület - Belső változók
          +-----------+
          |  PROG     |   BASIC RAM terület - BASIC program
          +-----------+
          |  SCALAR   |   BASIC RAM terület - BASIC skaláris
          |           |       változók
          +-----------+
          |  ARRAY    |   BASIC RAM terület - BASIC tömbök
          +-----------+
          |   --      |   BASIC RAM terület - Üres hely
          +-----------+
          |  STACK    |   BASIC RAM terület - STACK
          +-----------+
          |  STRING   |   BASIC RAM terület - STRING memória
          |           |       (ld. CLEAR parancs)
X'E800'   +-----------+
          |  SCREEN   |   HARDWARE terület - Képernyő puffer.
          |           |       32k/16k RAM esetén
          |           |       X'A800'/X'6800' a cim. Mérete
          |           |       alapértelmezésben 6 kbyte. A cim
          |           |       változtható X'C800'/X'8800' -ra.
          +-----------+
```

4.1 DCB - Device Control Block

A DCB a SYSRAM legfontosabb vezérlőblokkja. Cimét az IX regiszter tartalmazza, de a ROM X'1B' ciméről is ki lehet olvasni. A mostani verzióban a DCB cime X'4042'. A DCB más helyre még kerülhet, de a felosztása már nem fog változni. Kiosztása:

+0 (1) Utolsó klaviatúra karakter.

+1 (1) Klaviatúra repeat számláló.

+2.(1) Utolsó aktiv gomb cime. Az első két bit: X'80' - UPPER állapot, X'40' - nem volt még aktiv gomb. Az alsó hat bit a fizikai cim (ld. A. melléklet).

+3.(1) Klaviatúra/display flag

Bit 0 - DROW rutin - inverz vonalhúzás

Bit 1 - DROW rutin - ha bit0=0, akkor a vonalat 1=rajzolja, 0=törli

Bit 2 - 20 perces TIMER aktiv

Bit 3 - Funkció billentyű scan (speciális üzemmódban a 'lenyil' gomb le van nyomva)

Bit 4 - Printer X'1E', X'1F' konverzió tiltás. A két karakter a hosszú kis i, és a felfele nyil.

Bit 5 - az ernyőn levő cursor éppen világit

Bit 6 - Break tiltás - ha 1, akkor BRK(=1) helyett 0 kód jön be.

Bit 7 - Klaviatúra speciális üzemmód. Hatására a 'lenyil' shift gombként dolgozik, és a kódok értékéhez X'40'-et ad hozzá. A bitet az X'0035' - Linein funkció használja.

+4.(1) Klaviatúra időzités. (default=24, lehet 10-255)

+5.(1) Display flag

Bit 0 - Duplaszéles karakter

Bit 1 - Aláhúzás

Bit 2 - Inverz ernyő

Bit 3 - Inverz karakter

Bit 4 - Pre-clear

Bit 5 - Pre-clear puffer

Bit 6 - Inverz rajzolás

Bit 7 - Függőleges irás

+6.(1) A display kurrens X koordinátája. Ha V a karakterben mért koordináta, akkor X=4+V*6, ahol V=0-41. A rekesz kezdőértéke 4. (Mivel X irányban 0-255 pont van, és 256 nem osztható a karakterek 6 pont szélességével, ez egyenliti ki a differenciát.)

+7.(1) A display kurrens Y koordinátája. Ha W a sorban mért koordináta, akkor Y=1+W*12, ahol W=0-15. A rekesz kezdőértéke 1, mivel a karakterek első sora mindig üres (scrköz). Az Y értéke 0-191 lehet. A hardware cimzés a bal felső sarokból (0,0) indul.

*8.(1) Az ernyőpuffer (6 kbyte) kezdőcime. Mivel az alsó 11 bit értéke mindig 0, itt a cimnek csak a felső byte-ja tárolódik. Az ernyőpuffer cime csak két érték lehet: a memória utolsó 6 kbyte-ja, vagy egy ennél 8 kbyte-al kisebb érték. (Ennek hardware okai vannak, ugyanis a hardware ernyőpuffer átkapcsolás csak a cim egy bitjének állitására képes, a többi huzalozott.)

+9.(2) A 128 (X'80') feletti kódú karakterek generátortáblájának cime.

+11.(1) Pattern regiszter

+12.(1) Pattern puffer regiszter

+13.(1) Üres

+14.(1) A nyomtató maximális sorhossza.

+15.(1) A nyomtató maximális laphossza. Ha az itt megadott érték nagyobb, mint 127, nincs laphossz ellenőrzés.

+16.(1) A nyomtató soron belüli maradékszámlálója.

+17.(1) A nyomtató lapon belüli maradékszámlálója.

+18.(1) A nyomtató soron belüli poziciószámlálója.

+19.(1) Üres

+20.(1) IOSTAT byte a kazettakezelés részére.

Bit 0 - Nyitott kazettafile (1)

Bit 1 - Input(1) vagy output (0)

Bit 2 - A puffer üres (0), vagy kitöltve (1)

Bit 3 - TEST (1), vagy LOAD (0) funkció

Bit 4 - Hangadás (0), vagy nincs (1)

Bit 5 - Filenév van a kereséshez (1), vagy nincs

Bit 6 - Szövegirás az ernyőre (0), vagy nincs (1)

Bit 7 - Üres

+21.(1) IOCNTR byteszámláló, 0-tól 0-ig számol

+22.(2) I/O puffer pointer, az első szabad byte-ra mutat.

+24.(2) I/O puffer kezdőcim

+26.(1) Magnó rekordtipus.

+27.(1) Magnó hibás rekord számláló. értéke 0-ról indul, és BCD-ben növekszik. (azaz 0-99 lehet)

+28.(1) Magnó rekordszámláló. értéke 0-ról indul, és BCD-ben növekszik. (azaz 0-99 lehet)

+29.(1) Magnó fázisforditás jelző. Ha itt X'10' van, a magnó nem fordit fázist, ha X'EF', akkor igen.

+30.(1) Magnó időalapja.

+31.(1) Üres

4.2 Egyéb SYSRAM cimek

(000B) – Ígysram end+1
(0013) – Screen start
(001B) – DCR addr

(X'0023') - X'403D' (jelenleg) - REAL-TIME óra. Ez a három byte-os egész szám (legelső byte a legalacsonyabb helyiérték, a legutolsó a legmagasabb) bekapcsoláskor 0-ról indul, és minden 0.02 másodpercben (pontosabban 312*64 usec = 19.968 msec) 1-el növekszik. Az NMI kikapcsolása leállitja.

X'007E' - Generálási dátum, jelenleg X'84'

X'007F' - Verziószám, jelenleg X'01'
y'0180' – havahtun verzió szám

X'403E' - OUTBYTE. A hardware vezérlésére szolgál. Kiküldése a 0-63 cimek közül bármelyikre történhet. Mivel a hardware-ből visszaolvasni nem lehet, ezért itt van a parallel nyilvántartás. Kiosztását ld. az Input/Output leirásban. Induláskor értéke 0-ra áll be.

Az RST utasitások valamennyien a RAM-ba ugranak ki, funkciójuk elvégzése előtt. A megadott cimeken egy-egy JP utasitás található, amely tovább ugrik a funkciórutinra. Az egyes RST utasitások cime és funkciója:

RST 00H - Bekapcsolási törlés. Az X'000D' -t hivja.

RST 08H - X'4000' - BASIC

RST 10H - X'4003' - BASIC

RST 18H - X'4006' - BASIC

RST 20H - X'4009' - BASIC

RST 28H - X'400C' - Szabadon felhasználható

RST 30H - X'400F' - Szabadon felhasználható

RST 38H - X'4012' - Fenntartva az IM1-es IT-knek. Jelenleg itt jelenik meg a COMMODORE BUS Serv. Req jele, és a magnó input a 0-ba váltás esetén. (A COMMODORE-hoz be kell kötni egy opcionális diódát)

Egyéb ugrási cimek:

RESET - X'401E' - A RESET gomb a SYSRAM és a STACK initje után az itt található JP utasitásra ugrik. Igy a program le tudja kezelni a RESET gombot, de a megszakitás helyére visszatérni nem lehet a STACK initje miatt. A RESET-et hatálytalanitja az NMI letiltása (ld. OUTBYTE), de ez a REAL-TIME órát is kikapcsolja.

SSPEC - X'4027' - Speciális klaviatúra üzemmódban, ha a 'lenyil' + karakter sorozatot nyomjuk, kód+X'40' generálódik, majd ugrás történik az adott cimre. Itt egy JP utasitás van egy return-re. Igy lehetőség van PF (Prog.Func) gombok definiálására.

A jelenleg fel nem használt RST utasitások RAM rekeszeiben JP 3148H található (ugrás egy RET utasitásra), mig a többiben a megfelelő rutinra ugrás szerepel. (később ez JP 148H-ra változik)

NMI vége X'4018' HL és A a stacken

POP HL
PUSH AF
RETN

4.3 BASRAM cimek

X'40A2' - utolsó basic sornám (lm parancsnéd, kurblan &FFFF)

X'40AC' - STRING memória eleje. A STACK kezdőcime = (X'40A0')-2

X'40A4' - BASIC programok kezdőcime. Ha átirjuk, a tár elejéből kapunk más célra (pl. assembler programok) felhasználható területet. Átirása esetén ki kell adni a NEW parancsot, azaz a bentlevő programot törölni kell.

X'40AA'-X'40AC' - BASIC RND függvény kezdőértéke. A kezdőérték 3 byte-os egész formájában tárolódik LSB,M,MSB sorrendben.

X'40B1' - BASIC STRING memória végcime. Ha átirjuk, a tár végéből kapunk más célra (pl. assembler programok) felhasználható területet. Átirása esetén ki kell adni a CLEAR parancsot, amely gondoskodik a BASIC STACK áthelyezéséről, és a STRING memória újra allokálásáról. (Vigyázat ! Csak akkor, ha a CLEAR-t paraméterrel adjuk ki !)

X'40D6' - STRING kurrens pointer.

X'40E8' - A BASIC minden funkciórutin hivás előtt ide menti az SP-t.

X'40F9' - BASIC skaláris változók kezdőcime. A BASIC program mentéséhez ennek tartalma lesz a végcim a kazettakezelő számára.

X'40FB' - BASIC tömbváltozók kezdőcime. Vigyázat ! ez a cim a programfutás során dinamikusan változik !

X'40FD' - BASIC tömbváltozók végcime+1. Ez adja meg a BASIC program utáni első szabad cimet. Vigyázat ! ez a cim a programfutás során dinamikusan változik !

X'40FF' - DATA pointer (ind: =(4044) arra dolgozó)

5. INPUT/OUTPUT

Az I/O lényegében két input és két output port-ból áll, továbbá ide számitható a képernyő kezelés is.

Az output cimek kiosztása:

B'00xxxxxx' (0-63) - kezdőérték 0 (ld. még SYSRAM X'403B').

Kiosztása:

Bit 0 - NMI engedélyezés (1)/tiltás(0). Letiltott NMI esetén nincs REAL-TIME óra, és nem működik a RESET gomb.

Bit 1 - Joystick számláló léptetés (0-4 hasznos lekérdezés, 5-7 üres léptetés - ld. input 64-127)

Bit 2 - Magnó távvezérlés a tuchel 4. pontra, V24-nek is (± 5V) köthető.

Bit 3 - Hangszóró impulzus(1) ill. szünet (0)

Bit 4 - Display puffer vezérlés. (1) - magasabb cim (normal), (0) - alacsonyabb cim

Bit 5 - Magnó távvezérlés a tuchel 5. pontra, V24-nek is (± 5V) köthető. A bit a BUS A-22 pontján is megjelenik.

Bit 6-7 - Magnó illesztő vezérlés a tuchel 1. pontján. A két bit analóg összegzéssel jelenik meg, azaz 00 - 0 szint, 01/10 1 (közép) szint, az 11 pedig 2 (magas) szint. Igy lehet négyszöghullámot programozni.

B'01xxxxxx' (64-127) - kezdőérték 0. Kiosztása:

Bit 0 - üres

Bit 1 - üres

Bit 2 - COMMODORE Serial Clock

Bit 3 - COMMODORE Serial Data

Bit 4 - üres

Bit 5 - üres

Bit 6 - COMMODORE ATTN

Bit 7 - üres

Az input cimek kiosztása:

B'00xxxxxx' (0-63) A cim x-el jelölt része a klaviatúra 59 billentyűjét cimzi. (a cimeket ld. az A. mellékletben) A kapott input byte kiosztása:

Bit 0-1 - üres

Bit 2-3 - bővitésre fenntartva

Bit 4 - (1), ha az ernyő végén vagyunk

Bit 5 - magnó input a tuchel 3. pontjáról, ill. V24 input. A 0-ba átmenet megszakitást (RST 38H) okoz.

Bit 6 - Reset gomb be van nyomva (1) vagy nem (0)

Bit 7 - A cimzett klaviatúra gomb le van nyomva (1) vagy nem (0)

B'01xxxxxx' (64-127) A cim x-el jelölt része tetszőleges. a kapott input byte kiosztása:

Bit 0 - üres

Bit 1 - üres

Bit 2 - COMMODORE Serial Clock

Bit 3 - COMMODORE Serial Data

Bit 4 - COMMODORE Serv Req (RST 38H IT-t okoz, ha az opcionális dióda be van kötve)

Bit 5 - Joystick 2. A kurrens léptetéstől függően (ld. output 0-63) beolvassa a joystick mikrokapcsolóinak állását. Az öt léptetési állapotban a négy irány és a lövést jelző kapcsoló állását kapjuk. A megfelelő olvasáshoz a számlálót végig kell léptetni mind a nyolc állapoton.

Bit 6 - COMMODORE ATTN

Bit 7 - Joystick 1. (ld. bit 5.)

Ernyőkezelés:

Az ernyőkezeléshez a PRIMO a memória utolsó 16 kbyte-ját speciálisan figyeli. A processzor és az ernyőfrissités összeütközéseit mindig az ernyő javára oldja fel. Az ernyő mérete a hardware-ben jumperrrel állitható, vagy 6 kbyte (normál 192*256 => 42*16) vagy 8 kbyte (256*256 => 42*21). Az utóbbi bekötést a software nem támogatja, igy az ernyő tetején sötét csik marad. Átkötés esetén az ernyő cime X'ECCC' (ill. X'A000'/X'6000'), és gondoskodni kell a BASIC végcim megfelelő átirásáról. Az alternativ puffercim is változik X'C000'/X'8000' -ra (Az A16 változatnál a képernyő nem váltható).

6. KARAKTERGENERÁTOR VEZÉRLÉS

A 128 (X'80') feletti karakterek képét meg lehet változtatni. Ehhez saját táblát kell összeállitani, és címét a DCB+9, DCB+10 mezőbe kell helyezni (alacsony helyiérték elöl). Az egy karakter által elfoglalt mező képe 6*12-es:

Normál

```
+------+
|      |
|   *  |
|*****|
|*     |
|*     |
|***   |
|*     |
|*     |
|*     |
|*****|
|      |
|      |
|      |
+------+
```

Süllyesztett

```
+------+
|      |
|      |
|      |
|      |
|      |
|*    *|
|*    *|
|*    *|
| ****|
|     *|
|*    *|
| ***  |
|      |
+------+
```

A karakterdefinició általában 6*8-os méretű. A legfelső sort nem lehet átirni, azt a karaktergenerátor mindig üresen hagyja. A 8 sorból az elsőt az ékezetre, a hat oszlopból az utolsót pedig a betüközre szokás fenntartani. Az un. süllyesztett karaktereket külön meg kell jelölni, és azokat a karaktergenerátor 2 pozicióval lejjebb süllyeszti (ld. az ábrát). A karakter képét általában 8 byte irja le, ezek felépitése:

Byte 0 bit 0 Ha értéke 1, akkor a karakter süllyesztett.

bit 1-6 A karakter első sora. Ahol 1 érték van, ott lesz fénypont.

bit 7 Ha egy, a karakter végét jelzi.

Byte n bit 0 értéke közömbös.

bit 1-6 A karakter n.-dik sora. Ahol 1 érték van,

ott lesz fénypont.

bit 7 Ha egy, a karakter végét jelzi.

Megjegyezzük, hogy ha egy karakterleirás hosszabb, mint 8 byte, többsoros karaktert definiálunk, amely akár több karaktersoron (n*12 bitsoron) át is húzódhat. Ennek azonban több karakterkód elfoglalása az ára, mert ha a leirás n*8 hosszú, a kezdőkódot követő n-1 kód nem használható.

7. KAZETTAFORMÁTUM

A kazetta file három részből áll: header, adat(ok), és trailer. A file lehet szabványcs, ekkor a tipusokat szigoruan be kell tartani, és load paranccsal, illetve BASIC-ből a file olvasható lesz. A nem szabványos tipusok csak speciális assembler programmal olvashatóak. Két alaptipus a program és az adatfile. Az adatfile nem keveredhet semmi mással, a programfile azonban vegyesen tartalmazhat BASIC és Assembler programot, valamint elmentett ernyőképet (SCREEN). Az egyes blokkok felépitése:

Header 512 db. X'AA' - file szinkron

96 db. X'FF' + 3 db. X'D3' - rekord szinkron

1 byte rekordtipus

- X'83' - programfile

- X'87' - adatfile

1 byte rekordsorszám - értéke 0

1 byte névhossz - n=1-16

n byte név

1 byte kontrolösszeg - a rekordsorszámtól kezdve, és azt beleértve 8 bites összeg

Data 96 db. X'FF' + 3 db. X'D3' - rekord szinkron

1 byte rekordtipus

- X'F1' - BASIC program

- X'F5' - SCREEN

- X'F9' - Assembler

- X'F7' - adatfile

1 byte rekordsorszám (modulo 100) BCD-ben

2 byte töltési cim (alacsony/magas sorrendben) - BASIC programok és SCREEN esetén mindig relativ cim szerepel, amely 0-tól indul. Az adatfile-ok minden blokkja 0-ra van cimezve. Az assembler programok a valós betöltési cimet tartalmazzák.

1 byte hossz (0 megadása 256-ot jelent)

n byte adat (1-256) - Az adatfile-okban az elemek vesszővel vagy X'0D' (CR) karakterekkel vannak elválasztva a PRINT# utasitásnak megfelelően. Az egyes elemek átcsoroghatnak a blokkhatárokon.

1 byte kontrolösszeg - a rekordsorszámtól kezdve, és azt beleértve 8 bites összeg

Trailer 96 db. X'FF' + 3 db. X'D3' - rekord szinkron

1 byte rekordtipus

- X'B1' - BASIC vagy Assembler program
- X'B5' - SCREEN,
- X'B9' - autostartos Assembler program
- X'B7' - adatfile

1 byte rekordsorszám (modulo 100) BCD-ben

2 byte inditási cim (alacsony/magas sorrendben) - csak X'B9' tipus esetén.

1 byte kontrolösszeg - a rekordsorszámtól kezdve, és azt beleértve 8 bites összeg

8. ASSEMBLERBŐL HIVHATÓ RUTINOK

8.1 Display kezelés

```
+---------+
|         |
| DISPLAY |
|         |
+---------+
```

Display X'0015' Egy karakter írása a képernyő kurrens pozíciójába. Utána a kurrens pozició módosul (ld. DCB+6,7). A rutin az ernyővezérlő kódokat értelmezi. Minden regisztert ment.

Input: A= kiirandó karakter

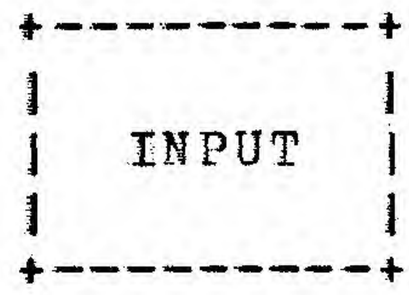
Output: A= változatlan, F= beáll - (OR A).

```
+---------+
|         |
| INKEYE  |
|         |
+---------+
```

INKEYE X'001D' Egy karakter olvasása a klaviatúráról várakozás nélkül. A kurrens pozició nem módosul, a karakter nem iródik vissza. Valamennyi kód bejön, beleértve a BRK-t is. Ha a klaviatúrán nincs semmi lenyomva, 0-t ad. Minden regisztert ment.

Input: --

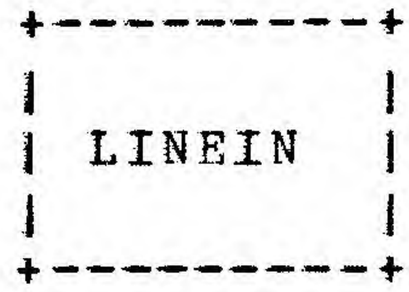
Output: A= karakter vagy 0, F= beáll - (OR A).



INPUT X'0025' Egy karakter olvasása a klaviatúráról várakozással. A kurrens pozició nem módosul, a karakter nem iródik vissza. Valamennyi kód bejön, beleértve a BRK-t is. A kurrens pozicióba kiirja a CURSOR-t, és vár. Elegendő sürüséggel hivva a REPEAT funkció is működik. Minden regisztert ment.

Input: --

Outpput: A= karakter, F= beáll - (OR A).



Lineir X'0035' Egy sor olvasása a klaviatúráról visszairással és sorszerkesztéssel. A funkció végét a max. 210 karakter beérkezése, a RETURN, illetve a BRK okozzák. A sor végét a puffer-ben egy nulla byte jelzi. BRK megnyomása esetén mindig üres sort kapunk. A vezérlőkódok értelmeződnek, de nem kerülnek a puffer-be. A 'lefele nyil' speciális SHIFT-ként dolgozik, a vele együtt lenyomott billentyűk a szokásos kódjuk helyett ennél X'40'-nel nagyobb kódot adnak be. A már beirt sort törölni lehet a CLS gombbal, ill. a SHIFT <-- gombbal. A balra nyil egy karakter tör-

lését okozza. A kurrens pozicióba kiirja a CURSOR-t, és vár a következő karakterre. A kurrens pozició a beirásnak megfelelően módosul. A REPEAT funkció működik. Minden regisztert ment.

Input: HL= puffercim - 1

Output: HL= változatlan, A= X'0D', F= CY, ha BRK volt.

```
+----------+
|          |
|  DCURS   |
|          |
+----------+
```

DCURS X'0080' Cursor cimzés rutin. Az átadott paraméterek-nek megfelelően beállitja a kurrens poziciót a DCB+6,7-ben.

Input: D= 0-15 sorcim, E= 0-41 pozició

Output: Az A, F -et rontja

```
+----------+
|          |
|   SET    |
|          |
+----------+
```

SET X'0083' A megcimzett ernyőpontot fehér alapnál eloltja, sötét alapnál kigyújtja. A cimzésnél a bal alsó sarok a (0,0). Ha az y>191, akkor semmit sem csinál, és CY feltételt ad.

Input: D= y (0-191), E= x (0-255)

Output: D= 191-y, F= CY érvénytelen y-nál, A, C romlik.

```
+---------+
|         |
|  RESET  |
|         |
+---------+
```

RESET X'0086' A megcimzett ernyőpontot sötét alapnál eloltja, fehér alapnál kigyújtja. A cimzésnél a bal alsó sarok a (0,0). Ha az y>191, akkor semmit sem csinál, és CY feltételt ad.

Input: D= y (0-191), E= x (0-255)

Output: D= 191-y, F= CY érvénytelen y-nál,

A, C romlik.

```
+---------+
|         |
|  POINT  |
|         |
+---------+
```

POINT X'0089' A megcimzett ernyőpontot megvizsgálja. A fehér alapon sötét ill. sötét alapon fehér pont esetén jelez 'igaz' feltételt. Igaz feltétel esetén nem zérus, egyébként zérus feltétel áll be az F regiszterben. A cimzésnél a bal alsó sarok a (0,0). Ha az y>191, akkor semmit sem csinál, és CY feltételt ad.

Input: D= y (0-191), E= x (0-255)

Output: D= 191-y, F= CY érvénytelen y-nál,Z

a hamis feltételnél, A, C romlik.

```
+---------+
|         |
|   POS   |
|         |
+---------+
```

POS X'008C' Visszaadja a kurrens soron belüli karakter-
poziciót.

Input: --

Output: A= pozició (0-41)

8.2 Printer kezelés

```
+---------+
|         |
|  PRINT  |
|         |
+---------+
```

Print X'002D' Egy karakter irása a printer kurrens pozici-
ójába. Utána a kurrens pozició módosul (ld. DCB+16,17,18). A kimenő vezérlőkódok közül az X'0D', X'0A' - soremelés, és az X'0C' - lapdobás értelmeződik. A többi vezérlőkód változtatás nélkül kiiródik. Az X'1E', X'1F' kódok (hosszú kis i, felfele nyil) a DCB+3 szerint konvertálódhatnak. Minden regisztert ment. A nyomtatón a helyes működés érdekében a svéd karakterkészletet kell beállitani.

Input: A= kiirandó karakter

Output: A= változatlan, F= beáll A szerint
(OR A).

8.3 Ernyőpuffer kezelés

```
+---------+
|         |
|  ALLOC  |
|         |
+---------+
```

Alloc X'3123' Második ernyőpuffert allokál a buff-8kbyte cimre. Az allokálást csak 16kbyte RAM-nál nagyobb gépben végzi el. Ezt a területet a BASIC-től elveszi, igy kiadása után CLEAR parancs szükséges. A két ernyőpuffer közötti 2kbyte mezőt tetszőleges célra fel lehet használni.

Input: --

Output: A,F elromlik.

```
+---------+
|         |
| SWITCH  |
|         |
+---------+
```

Switch X'312D' Csak már allokált második ernyőpuffer esetén működik. Kivárja az ernyőfrissités végét, majd mind a hardware-ben, mind a DCB+8 -ban átirja az ernyőpuffer cimét. A funkció oda-vissza kapcsol a két ernyőpuffer között.

Input: --

Output: A,F elromlik.

```
+----------+
|          |
|  WRHDR   |
|          |
+----------+
```

Wrhdr X'0092' File header irására szolgál. A DCB+26 -ba hivás előtt be kell irni a header tipust. Komplett headert ir ki. A HL -ben kell átadni a file név leiró cimét. Ennek felépitése megegyezeik a standard string leirással (első byte a hossz, második és harmadik byte a string cime, alacsony/magas helyiérték sorrendben). Elinditja a magnót, felvesz 4 sec szünetet, majd felirja a header-t.

Input: HL= névleirás, DCB+20= hangadás, DCB+26= tipus

Output: HL, DCB+26 változatlan, a DCB+28= 0 (rekordszám) AF, BC -t ront.

```
+----------+
|          |
| WRDATA   |
|          |
+----------+
```

Wrdata X'0095' File adat irására szolgál. A DCB+26 -ba hivás előtt be kell irni az adattipust. Komplett adatot ir ki, a DCB+28 (rekordsorszám) automatikusan módosul. Egyszerre akárhány blokk kiirható vele. A file-ba töltési cimként a memóriacim-offset érték kerül. BASIC program kimentésénél offset = kezdő-

cim.

Input: DE= kezdőcim, HL= végcim+1, BC= off-
set, DCB+20= hangadás, DCB+26=
tipus, DCB+28= rekordsorszám (BCD)

Output: DCB+28 inkrementálódik a kiirt blok-
koknak megfelelően. AF, DE, HL rom-
lik

```
+---------+
|         |
|  WRTRL  |
|         |
+---------+
```

Wrtrl X'009B' File trailer irására szolgál. A DCB+26 -ba hivás előtt be kell irni a header tipust. Komplett trailert ir ki, majd leállitja a motort. Automatikus indítású programhoz nem használható, mert az inditási címet nem tud-ja felirni. Ilyen esetben byte-onkénti kivi-telt kell használni. A rekordsorszámot növe-li.

Input: DCB+20= hangadás, DCB+26= tipus,
DCB+28= rekordsorszám (BCD)

Output: DCB+28 megnövelve. AF, BC romlik

```
+---------+
|         |
|  WRSYN  |
|         |
+---------+
```

Wrsyn X'3ADD' File szinkron irására szolgál. Kiir 96 db. X'FF'-et, és 3 db. X'D3'-at.

Input: --

Output: --, AF, BC romlik

+---------+
| |
| WRBYTE |
| |
+---------+

Wrbyte X'3AF4' Egy karaktert ir a magnóra, kontrolösszeg számitás nélkül. A DCB+20 X'08' bitje (1/0) engedélyezi/tiltja a hangadást.

Input: A= kiirandó karakter, DCB+20= hangadás vezérlés

Output: --, AF romlik

+---------+
| |
| WRBYTC |
| |
+---------+

Wrbytc X'3AF0' Egy karaktert ir a magnóra, kontrolösszeg képzéssel. A DCB+20 X'08' bitje (1/0) engedélyezi/tiltja a hangadást. A kiirt karakter hozzáadódik a C regiszter tartalmához. Az első hivás előtt C-t nullázni kell, majd a további hivásokhoz meg kell őrizni. A blokk végére ki kell irni a Wrbyte funkcióval a C-ben levő kontrolösszeget. A rekordszám az első olyan byte, amit bele kell számolni a kontrolösszegbe.

Input: A= kiirandó karakter, C= előző kontrolösszeg, DCB+20= hangadás vezérlés

Output: C= módositott kontrolösszeg. AF romlik

```
+---------+
|         |
|  DRVON  |
|         |
+---------+
```

Drvon X'3B24' A magnómotor bekapcsolására szolgál.

Input: --

Output: AF-t rontja

```
+---------+
|         |
| DRVOFF  |
|         |
+---------+
```

Drvcff X'3B26' A magnómotor kikapcsolására szolgál.

Input: --

Output: AF-t rontja

```
+---------+
|         |
| SRCHDR  |
|         |
+---------+
```

Srchdr X'008F' File header keresésére szolgál. A magnót inditja, majd 20 msec csendet vár. Ezután keresi a file szinkront, időalapot mér, és megállapitja, hogy szükséges-e fázisfordi-tás. Ezután keresi a rekordszinkront, beol-vassa a headert, és ha volt név megadva, teszteli. Ha kell, üzenetet ir a displayra, ahol beállitja a pre-clear üzemmódot.

Input: DCB+20, DCB+26, HL= filenév leiró cime.

Output: DCB+27,29,30 beállitva

```
+----------+
|          |
|  SRCSYN  |
|          |
+----------+
```

Srcsyn X'3C75' Rekord szinkron keresésére szolgál. Ha találát, beolvassa a rekordtipust.

Input: --

Output: A= rekordtipus

```
+----------+
|          |
|  INBYTE  |
|          |
+----------+
```

Inbyte X'009E' Beolvas egy byte-ot, és növeli a kontrolösszeget.

Input: D= eddigi kontrolösszeg

Output: A= input byte, D= új kontrolösszeg

```
+----------+
|          |
|  CHKSUM  |
|          |
+----------+
```

Chksum X'3C22' Beolvassa a kontrolösszeget, és összehasonlitja a D-vel. Ha kell, növeli a hibaszámlálót, és kiirja a rekordszámmal együtt a display-re.

Input: D= eddigi kontrolösszeg, E= rekordsorszám, DCB+27

Output: DCB+27

```
+---------+
|         |
| RDDATA  |
|         |
+---------+
```

Rddata X'0098' File adat olvasására szolgál. Komplett blokkot olvas be. Előtte hivni kell a rekordszinkron keresést, majd be kell olvasni a rekordsorszámot. (ehhez D= 0 kell, mert a sorszám már beleszámit a kontrolösszegbe.) A DCB+27 automatikusan módosul. A blokk a memóriába a töltési cim + offset cimre kerül. A checksum-ot ellenörzi.

Input: D= előző kontrolösszeg (rekordsorszám) E= rekordsorszám (BCD), BC= offset, DCB+27

Output: DCB+27,28, HL=HL+BC, BC, DE romlik

8.5 Egyéb rutinok

```
+---------+
|         |
| PWRESET |
|         |
+---------+
```

PWRESET X'000D' POWER-ON reset rutin, nem tér vissza. Inicializálja a SYSRAM-ot, majd a BASIC init-et hivja.

Input: --

Output: -- nem tér vissza !

+----------+
| |
| RELOC |
| |
+----------+

Reloc X'002B' Később kerül implementálásra. Önáthelyező programok támogatására szolgál. Hivásakor megadja a hivó CALL utasitást követő cimet.

Input: --

Output: HL= a hivó CALL utáni cim.

+----------+
| |
| CLRS |
| |
+----------+

CLRS X'1B61' A BASIC CLEAR funkcióját hajtja végre a STACK átállitással együtt. Hivásához be kell állitani a STRING memória elejének, és végének cimeit a BASRAM-ban. (X'40A0', X'40B1', X'40D6')

Input: BASRAM rekeszek.

Output: BC-t rontja

+----------+
| |
| NEWC |
| |
+----------+

NEWC X'1B4A' A BASIC NEW funkcióját hajtja végre, ernyőtörléssel együtt.

Input: --

Output: --

+---------+
| |
| NEW |
| |
+---------+

NEW X'1B4D' A BASIC NEW funkcióját hajtja végre, ernyő-
törlés nélkül.

Input: --

Output: --

+---------+
| |
| CLEAR |
| |
+---------+

CLEAR X'1E83' A BASIC CLEAR funkcióját hajtja végre a
STACK átállitással együtt.

Input: DE= stringterület új hossza

Output: BC-t rontja

+---------+
| |
| VARPTR |
| |
+---------+

VARPTR X'2664' BASIC skaláris változók címének megkeresésé-
re szolgál. Alapos tesztelést kiván, használata bizonytalan. Még nem allokált változó esetén az allokálást is elvégzi.

Input: BC= név, D= tipus (2= INT, 3= STR,
4= SNG, 8= DBL)

Output: DE= a változó cime

BEEP

BEEP X'3F68' A BASIC BEEP rutint hivja. Leirását ld. ott.

BC= időtartam, DE= T/2

Output: AF, BC romlik

RUN

RUN X'1B5D' A BASIC programot inditja. Nem tér vissza a hivóhoz. A rutint nem CALL, hanem JP utasitással hivjuk, és hivás előtt a STACK-re kell tenni X'1D1E'-t.

Input: STACK= X'1D1E'

Output: -- , nem tér vissza.

LOAD

LOAD X'050F' A BASIC LOAD rutinját hivja. Ha a program automatikusan indul, nem tér vissza a hivóhoz. Hiváskor a nevet BASIC forrássor formájában kell megadni, X'0D'-vel zárva. Ha a nevet nem adjuk meg, egy üres soremelést kell átadni.

Input: (HL)-> "név",X'0D'

Output: ??? romlik

+----------+
| |
| TEST |
| |
+----------+

TEST X'059B' A BASIC TEST rutinját hivja. Hiváskor a nevet BASIC forrássor formájában kell megadni, X'0D'-vel zárva. Ha a nevet nem adjuk meg, egy üres soremelést kell átadni.

Input: (HL)-> "név",X'0D'

Output: ??? nem tér vissza

+----------+
| |
| SAVE |
| |
+----------+

SAVE X'04CD' A BASIC SAVE rutinját hivja. Hiváskor a nevet BASIC forrássor formájában kell megadni, X'0D'-vel zárva.

Input: (HL)-> "név",X'0D'

Output: ??? nem tér vissza

A. melléklet - klaviatúra fizikai cimek

Klaviatúra cimek kód szerint rendezve

0	0	Y	23	17	7 /	43	2B	. :
1	1	▲	24	18	H	44	2C	M
2	2	S	25	19	SPACE	45	2D	9)
3	3	SHIFT	26	1A	B	46	2E	I
4	4	E	27	1B	6 &	47	2F	, ;
5	5	UPPER	28	1C	G	48	30	Ü
6	6	W	29	1D	5 %	49	31	' *
7	7	CTR	30	1E	V	50	32	P
8	8	D	31	1F	4 €	51	33	Ű ű
9	9	3 #	32	20	N	52	34	O
10	A	X	33	21	8 (	53	35	CLS
11	B	2 "	34	22	Z	55	37	RETURN
12	C	Q	35	23	+ ?	57	39	<--
13	D	1 !	36	24	U	58	3A	É
14	E	A	37	25	0 =	59	3B	Ó Ő
15	F	▼	38	26	J	60	3C	Á
16	10	C	39	27	> <	61	3D	-->
18	12	F	40	28	L	62	3E	Ö
20	14	R	41	29	- i	63	3F	BRK
22	16	T	42	2A	K			

Klaviatúra cimek karakter szerint rendezve

. :	43	2B	C	16	10	T	22	16
<--	57	39	D	8	8	U	36	24
+ ?	35	23	E	4	4	Ü	48	30
▲	1	1	É	58	3A	Ű ű	51	33
▼	15	F	F	18	12	V	30	1E
- i	41	29	G	28	1C	W	6	6
-->	61	3D	H	24	18	X	10	A
, ;	47	2F	I	46	2E	Y	0	0
> <	39	27	J	38	26	Z	34	22
' *	49	31	K	42	2A	0 =	37	25
BRK	63	3F	L	40	28	1 !	13	D
CLS	53	35	M	44	2C	2 "	11	B
CTR	7	7	N	32	20	3 #	9	9
RETURN	55	37	O	52	34	4 €	31	1F
SHIFT	3	3	6 &	59	3B	5 %	29	1D
SPACE	25	19	Ö	62	3E	6 &	27	1B
UPPER	5	5	P	50	32	7 /	23	17
A	14	E	Q	12	C	8 (	33	21
Á	60	3C	R	20	14	9)	45	2D
B	26	1A	S	2	2			

Karakterkészlet

0	0			51	33	3		102	66	f
1	1	CTL-A,	BRK	52	34	4		103	67	g
2	2	CTL-B		53	35	5		104	68	h
3	3	CTL-C		54	36	6		105	69	i
4	4	CTL-D		55	37	7		106	6A	j
5	5	CTL-E		56	38	8		107	6B	k
6	6	CTL-F		57	39	9		108	6C	l
7	7	CTL-G,	BELL	58	3A	:		109	6D	m
8	8	CTL-H,	<--	59	3B	;		110	6E	n
9	9	CTL-I,	SH -->	60	3C	<		111	6F	o
10	A	CTL-J,	▼	61	3D	=		112	70	p
11	B	CTL-K		62	3E	>		113	71	q
12	C	CTL-L,	CLS	63	3F	?		114	72	r
13	D	CTL-M,	RETURN	64	40	é	(U)	115	73	s
14	E	CTL-N,	CR	65	41	A	(U)	116	74	t
15	F	CTL-O		66	42	B	(U)	117	75	u
16	10	CTL-P		67	43	C	(U)	118	76	v
17	11	CTL-Q		68	44	D	(U)	119	77	w
18	12	CTL-R		69	45	E	(U)	120	78	x
19	13	CTL-S		70	46	F	(U)	121	79	y
20	14	CTL-T		71	47	G	(U)	122	7A	z
21	15	CTL-U		72	48	H	(U)	123	7B	ő
22	16	CTL-V		73	49	I	(U)	124	7C	ö
23	17	CTL-W		74	4A	J	(U)	125	7D	á
24	18	CTL-X,	SH <--	75	4B	K	(U)	126	7E	ü
25	19	CTL-Y,	-->	76	4C	L	(U)	127	7F	ű
26	1A	CTL-Z,	SH ▼	77	4D	M	(U)	128	80	Pi
27	1B			78	4E	N	(U)	129	81	sz. (
28	1C	CTL-Ö		79	4F	O	(U)	130	82	sz.)
29	1D	CTL-Á		80	50	P	(U)	131	83	<- (K)
30	1E	CTL-Ü,	↑	81	51	Q	(U)	132	84	-> (K)
31	1F	▲		82	52	R	(U)	133	85	▼ (K)
32	20	SPACE		83	53	S	(U)	134	86	Gyök
33	21	!		84	54	T	(U)	135	87	Pipa
34	22	"		85	55	U	(U)	136	88	Integrál
35	23	#		86	56	V	(U)	137	89	Ft
36	24	$		87	57	W	(U)	138	8A	|
37	25	%		88	58	X	(U)	139	8B	Ü
38	26	&		89	59	Y	(U)	140	8C	Paragraph
39	27	'		90	5A	Z	(U)	141	8D	‡
40	28	(		91	5B	ő		142	8E	Á
41	29	)		92	5C	Ö	(U)	143	8F	
42	2A	*		93	5D	Á	(U)	144	90	
43	2B	+		94	5E	Ü	(U)	145	91	Hullám
44	2C	,		95	5F	ú		146	92	Omega
45	2D	-		96	60	é		147	93	Mü
46	2E	.		97	61	a		148	94	Négyzet
47	2F	/		98	62	b		149	95	Azonos
48	30	0		99	63	c		150	96	Szumma
49	31	1		100	64	d		151	97	Produktjel
50	32	2		101	65	e				

A (K) a nyilak nyomtatási kódját jelzi. Az (U) a nagybetűk mellé került a táblázatban.

Ernyővezérlő kódok

0	0		A BASIC string végjelnek használja, kiirva nincs poziciótovábbitás
1	1	CTL-A, BRK	BASIC programfutás megállitás, kiirva alaphelyzetbe állitja a képernyőkezelő opciókat.
2	2	CTL-B	Duplaszéles karakterek irása (21 char/scr)
3	3	CTL-C	Az ernyő alapszine fehér lesz, és a karakterek feketén jelennek meg.
4	4	CTL-D	A karakter háttérszine az ernyő ellentettje lesz, a karakter ehhez képest inverzen jelenik meg.
5	5	CTL-E	A karakterek aláhúzva jelennek meg.
6	6	CTL-F	A karakterek helye kiirás előtt törlődik
7	7	CTL-G, BELL	Megszólaltatja a dudát
8	8	CTL-H, <--	Kurrens pozició törlése visszalépéssel
9	9	CTL-I, SH -->	Tabulátor (16 poziciónként), illetve BASIC EDIT esetén az egész sor végigirása RETURN-el
10	A	CTL-J, ▼	Outputként hatástalan, inputként BASIC EDIT ill. Linein esetén speciális SHIFT, a kódok értékéhez X'40'-et ad hozzá
11	B	CTL-K	
12	C	CTL-L, CLS	Ernyőtörlés
13	D	CTL-M, RETURN	Uj sor, BASIC EDIT esetén a maradék sor törlődik
14	E	CTL-N, CR	~~Uj~~ sor eleje (ugyanabban a sorban)
15	F	CTL-O	Függöleges irás. Az ernyő mérete ilyenkor 32*21, és a PRINT@ parancs koordinátái meg vannak cserélve.
16	10	CTL-P	Subscript - a következő kiirás 4 pontsorral lejjebb történik
17	11	CTL-Q	Superscript - a következő kiirás 4 pontsorral feljebb történik.
18	12	CTL-R	Duplaszéles char üzemmód kikapcsolás
19	13	CTL-S	Az ernyő alapszine sötét
20	14	CTL-T	A karakter alapszine az ernyővel egyező lesz
21	15	CTL-U	Aláhúzás kikapcsolás
22	16	CTL-V	Karakterpoziciók nem törlődnek kiirás előtt. (kompozit karakterek előállitása lehetséges)
23	17	CTL-W	Vizszintes irás (42*16)
24	18	CTL-X, SH <--	Linein funkciónál a beirt sor törlődik, BASIC EDIT esetén a sor editálása elölről kezdődik.
25	19	CTL-Y, -->	BASIC EDIT esetén a sor következő karaktere kiiródik az ernyőre

BASIC hibajelzések

1	1	NF	NEXT hiba, NEXT használata megelőző FOR nélkül
2	2	SN	Szintaktikus hiba. Ha a sor visszairódik, rögtön javitható
3	3	RG	RETURN használata megelőző GOSUB nélkül
4	4	OD	Több adat olvasása READ utasitással, mint amennyi a DATA utasitásokban meg van adva
5	5	FC	Függvényhivási (paraméterezési) hiba, vagy negativ tömbindex
6	6	OV	Túlcsordulás, az egyes ábrázolási pontosságok túllépése számolás közben
7	7	OM	Memóriahiány
8	8	UL	Definiálatlan cimkére hivatkozás
9	9	BS	Indexhatár vagy dimenziószám túllépés
10	A	DD	Tömb újradimenzionálása. Leggyakoribb oka vagy több DIM utasitás használata, vagy a DIM utasitás nem előzi meg az első hivatkozást. (Az első hivatkozáskor a tömb minden indexe 0-10 határra deklarálódik)
11	B	/0	Nullával osztás
12	C	ID	Az utasitás parancsként nem használható
13	D	TM	Adattipus hiba
14	E	OS	String memória túlcsordulás. A hiba javitásához a program elején meg kell adni a CLEAR nn utasitást, amely beállitja a String memória méretét. (alapértelmezés = 50)
15	F	LS	255 karakternél hosszabb string használata
16	10	ST	A megengedettnél bonyolultabb stringkifejezés használata
17	11	CN	A CONTINUE parancs illegális használata
18	12	NR	A RESUME parancs illegális használata
19	13	RW	A RESUME parancs érvénytelen
20	14	UE	Definiálatlan hiba
21	15	MO	Hiányzó operandus
22	16	FD	Hibás adat a file-ban, vagy korai filevég
23	17	L3	Floppydiszkes rendszer hibajelzése